VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

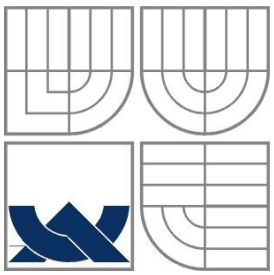
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ŠTRUKTÚRNY DESIGN POMOCOU CELULÁRNYCH AUTOMATOV

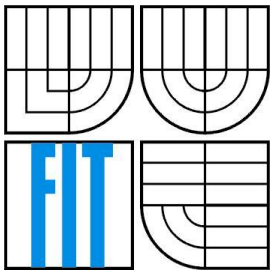
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JAKUB BEZÁK
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

STRUKTURNÍ DESIGN POMOCÍ CELULÁRNÍCH AUTOMATŮ

STRUCTURAL DESIGN USING CELLULAR AUTOMATA

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB BEZÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL BIDLO, Ph.D.

BRNO 2012

Abstrakt

Cílem tohoto textu je čtenáři co nejvíce přiblížit problematiku celulárních automatů, jejich návrhu a použití pro strukturní design. Na návrh automatů se nejčastěji používá genetických algoritmů, které jsou zde pro lepší pochopení také prezentovány. Jako struktury jsou použity řadicí sítě, které však nejsou součástí automatů, ale jsou generovány samostatně, pomocí pravidel lokální přechodové funkce automatu.

Abstract

The aim of this paper is to introduce the readers to the field of cellular automata, their design and their usage for structural design. Genetic algorithms are usually involved in designing complicated cellular automata, and because of that they are also briefly described here. For the purposes of this work sorting networks are considered as suitable structures to be designed using cellular automata, however, they are not a part of the automata but they are generated separately by modified rules of a local transition function.

Klíčová slova

Genetický algoritmus, celulární automat, řadicí síť, evoluční návrh.

Keywords

Genetic algorithm, cellular automaton, sorting network, evolutionary design.

Citace

Bezák, Jakub: Štruktúrny design pomocou celulárnych automatov, diplomová práce, Brno, FIT VUT v Brně, 2012

Štruktúrny design pomocou celulárnych automatov

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Bidla, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Jakub Bezák
23. 05. 2012

Poděkování

Týmto by som chcel poďakovať môjmu vedúcemu Ing. Michalovi Bidlovi, Ph.D. za jeho rady a podnety ako aj poskytnutú literatúru.

Obsah

1 Úvod

Azda každý človek sa aspoň raz vo svojom živote zamýšľal ako vznikla Zem a život na nej, ako sa na nej objavil človek. Tí, ktorí sa touto myšlienkou zaoberali hlbšie, mohli si položiť otázku prečo je každý človek iný, má inú farbu vlasov, očí, či pokožky. Ďalej sa mohli pýtať ako je možné, že niektoré živočíchy sa prispôsobili životu na púšti alebo za polárnym kruhom. Odpoveďou na väčšinu otázok je evolúcia, vďaka ktorej sa jednotlivé organizmy postupne vyvíjali a prispôsobovali prostrediu, v ktorom žili. Tento biologický proces je hlavnou inšpiráciou pre evolučné prehľadávacie algoritmy použité aj v tejto práci.

Prvé pokusy s výpočtami používajúcimi túto techniku sa objavili už v 50-tych rokoch minulého storočia [Hyn08], ale kvôli nízkemu výpočtovému výkonu boli pár desiatok rokov prehliadané. Až v posledných rokoch vzrástla popularita evolučného návrhu v mnohých vedeckých oblastiach, kde pomáha nájsť nové inovatívne riešenia, ktoré by nebolo možné dosiahnuť klasickými konvenčnými postupmi.

Jednou z oblastí je návrh celulárnych automatov za účelom dosiahnutia špecifickej funkcionality, ktorý by bol inak značne komplikovaný. Ako už názov práce napovedá, jej hlavnou témou budú práve celulárne automaty a ich využitie v štruktúrnom dizajne, konkrétne pri generovaní radiacich sietí.

Práca je rozdelená na 7 kapitol, v ktorých je popísané teoretické pozadie problematiky, sú formálne definované použité modely, navrhnuté riešenie pre počítanie so štruktúrami v automate a prezentované sú vybrané výsledky experimentov.

Kapitola 2 popisuje najskôr neformálne a potom formálne celulárne automaty. Ďalej prezentuje automat ako výpočtový model, ktorý je schopný sebareplikácie či už samého seba alebo štruktúr, ktoré obsahuje. Dočítame sa tiež o ďalších aplikáciách automatov a o triedach, podľa ktorých je možné automaty klasifikovať. Okrajovo sú spomenuté alternatívne modely automatov, ktoré sa neriadia zaužívaným konceptom. Samostatná časť je venovaná návrhu automatov pomocou genetických algoritmov a pomocou celulárneho programovania.

Kapitola 3 uvádza základné pojmy genetického algoritmu, jeho pôvod a krátku históriu ako aj samotný algoritmus a jeho jednotlivé kroky. Podrobnejšie sa dozvieme o spôsobe výberu chromozómov, ktoré sa podieľajú na reprodukcii o najčastejšie používaných genetických operátoroch – krížení a mutácii a o alternatívnych spôsoboch tvorby novej populácie v nasledujúcom kroku genetického algoritmu.

V kapitole 4 sa nachádza definícia radiacich sietí, princíp overenia správnosti fungovania radiacich sietí, ďalej je možné nájsť prehľad najlepších známych sietí a niektoré konvenčné spôsoby tvorby radiacich sietí. Ďalej je prezentovaná tvorba radiacich sietí pomocou celulárneho automatu za pomoci genetického algoritmu. Sú navrhnuté dva spôsoby kódovania (relatívne a absolútne) radiacich

sietí v rámci pravidiel lokálnej prechodovej funkcie celulárneho automatu ako aj reprezentácia týchto pravidiel v chromozóme genetického algoritmu. Tiež sa zaoberá myšlienkou priamej reprezentácie radiacej siete v 2D celulárnom automate.

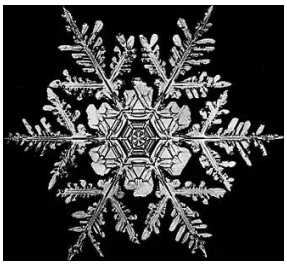
Kapitola 5 obsahuje hlavnú myšlienku zamerania práce, ktorou je tvorba postupne sa rozširujúcich radiacich sietí pomocou opakovania nájdených.

Kapitola 6 prináša prehľad vykonaných experimentov pre generovania postupne rozširujúcich sa radiacich sietí pri použití rôznych spôsobov opakovania vzorov. Každý experiment obsahuje tabuľky so štatistikami experimentu a obrázky s príkladmi vygenerovaných sietí.

Text je zhrnutý v kapitole 7, a sú tu ponúknuté možnosti ďalšej práce.

2 Celulárne automaty

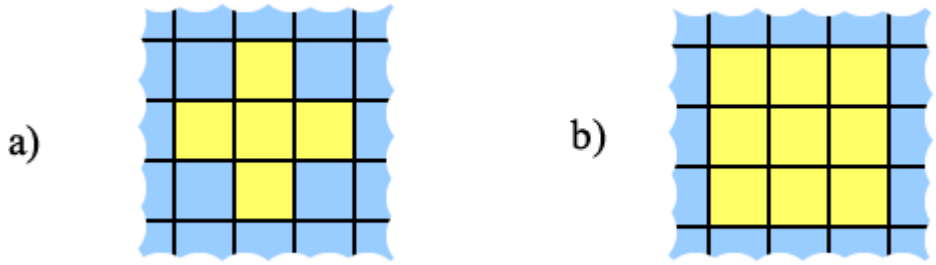
Celulárny automat je ďalší zo systémov používaných v informatike, ktorý našiel inšpiráciu v prírode. Hlavný princíp spočíva vo využití princípu ontogenézie (angl. development) spočívajúcej v simulácii vývoja mnohobunkových organizmov v prírode. Tieto procesy zahŕňajú tiež javy ako emergencia (samoorganizácia) alebo emergentné správanie, kedy vznikajú komplexné systémy alebo vzory na základe opakovania lokálnych interakcií bez nejakej centrálnej autority alebo hlavnej riadiacej jednotky. Príklady takýchto procesov môžeme pozorovať napríklad pri tvorbe snehových vločiek, pri pohybe molekúl, v terciálnej štruktúre DNA, správaní kolónií mravcov alebo včiel, pri pohybe kŕdľa vtákov, atď. Jeden z príkladov uvádza obrázok 1.



Obrázok 1: *Príklad emergencie, komplexný symetrický vzor snehovej vločky.*

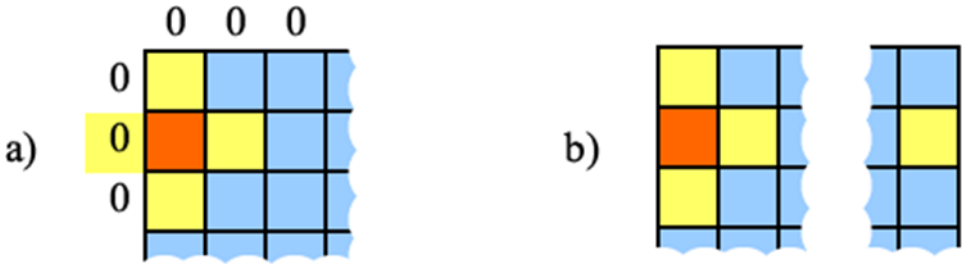
Ako prví sa myšlienkou celulárneho automatu zaoberali Ulam a von Neumann v štyridsiatych rokoch dvadsiateho storočia [vN66]. Celulárny automat si môžeme predstaviť ako *n-rozmerné* pole buniek usporiadaných do pravidelnej mriežky, kde *n* je kladné celé číslo, pričom najčastejšie sa používajú hodnoty 1 a 2 a každá bunka sa môže v danom okamžiku nachádzať v jednom z konečného počtu stavov. Stavy jednotlivých buniek sa menia synchrónne v každom kroku automatu. Každá bunka zmení stav na základe svojho aktuálneho stavu a stavov buniek v jej okolí. Súbor pravidiel na zmenu stavu bunky sa nazýva lokálnej prechodová funkcia. Ak je táto funkcia identická pre všetky bunky celulárneho automatu, nazývame automat uniformný, v opačnom prípade neuniformný. Celulárne automaty pracujú s diskrétnym časom a priestorom.

Okolie bunky (*cellular neighborhood*) obvykle tvoria priľahlé bunky. Pri jednorozmernom automate uvažujeme *r* buniek pripojených z každej strany ako aj bunku samotnú, okolie teda tvorí *2r +1* buniek, kde parameter *r* sa nazýva rádius. U dvojrozmerných automatov uvažujeme typicky 2 druhy okolí. Prvým je 5-okolie, kedy počítame s bunkami, ktoré zdieľajú s aktuálnou bunkou jednu spoločnú hranu, 9-okolie okrem týchto buniek využíva aj bunky na diagonálach, ktoré sú s aktuálnou bunkou spojené práve v jednom vrchole. 5- a 9-okolie ilustruje obrázok 2.



Obrázok 2: *Ukážka a) 5-okolia a b) 9-okolia bunky celulárneho automatu.*

Stav všetkých buniek v čase *t* označujeme ako konfigurácia celulárneho automatu, postupnosť týchto konfigurácií chápeme ako výpočet (vývoj) celulárneho automatu. Pre celulárne automaty konečnej veľkosti musíme zvoliť okrajové podmienky, a to buď nulové, kedy bunkám z okolia bunky, ktoré by sa nachádzali mimo celulárneho automatu, priradíme stav 0, alebo cyklické, čo znamená, že sa v každej dimenzii pozeráme na mriežku ako na toroid a bunky najviac vľavo (hore) a najviac vpravo (dole) sa tvária ako susedné. Nulové aj cyklické podmienky sú znázornené graficky na obrázku 3.



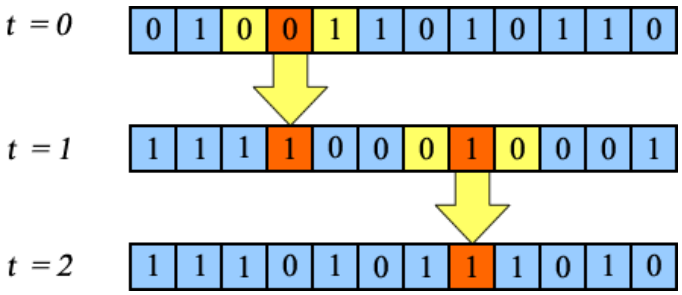
Obrázok 3: *Ukážka a) nulových a b) cyklických okrajových podmienok pre 5-okolie v 2-dimenzionálnom celulárnom automate.*

Ako príklad celulárneho automatu uvedieme 1-rozmerný, binárny (2-stavový) automat o veľkosti 12 buniek, kde lokálna prechodová funkcia bude funkcia parita (pravidlo XOR). Pre okolie bunky budeme uvažovať parameter $r = 1$ a cyklické okrajové podmienky. Dostávame 8 pravidiel, ktoré sú uvedené v tabuľke 1.

$S_0S_1S_2$	$S_{next} = S_0 \oplus S1 \oplus S2$	$S_0S_1S_2$	$S_{next} = S_0 \oplus S1 \oplus S2$
000	0	100	1
001	1	101	0
010	1	110	0
011	0	111	1

Tabuľka 1: *Lokálna prechodová funkcia pre ukážkový celulárny automat.*

Dva kroky výpočtu celulárneho automatu na základe pravidiel z tabuľky Tabuľka **1** ukazuje obrázok 4.



Obrázok 4: *Dva kroky výpočtu celulárneho automatu, využívajúceho pravidlá z Tabuľky 1.*

Okrem prezentovaného modelu synchrónneho deterministického uniformného automatu existuje niekoľko ďalších variant, ktoré majú svoje využitie. Jednotlivé varianty menia práve spomínané parametre automatu. Neuniformný celulárny automat nepracuje s jednou lokálnou prechodovou funkciou, ale každá jeho bunka má vlastnú sadu pravidiel. Nedeterministický celulárny automat, na rozdiel od deterministického, môže pre rovnakú počiatočnú konfiguráciu dosiahnuť rôzne výsledky, pretože nasledujúci stav jeho buniek bude s určitou pravdepodobnosťou *p* hodnota *x* a s pravdepodobnosťou *(1-p)* hodnota *y*. Iné automaty dokonca pracujú asynchrónne, čo však komplikuje propagáciu informácie naprieč bunkami automatu a tým pádom sťažuje jeho analýzu [Wol94]. V neposlednom rade, je možné zmeniť architektúru automatu a napríklad každú bunku spojiť s identickým počtom iných buniek, ktoré nemusia byť v spomínanom 5- a 9-okolí [Sip97b].

2.1 Formálna definícia

Moshe Sipper a Marco Tomassini vo svojej práci [ST98] formálne definujú celulárny automat nasledovne:

d-dimenzionálny celulárny automat pozostáva z konečnej alebo nekonečnej *d*-dimenzionálnej mriežky buniek. Každá bunka nadobúda hodnoty stavov z konečnej, typicky malej množiny celých čísel. Hodnota každej bunky v čase *t* je funkciou hodnôt v lokálnom okolí buniek v čase $t - 1$. Bunky aktualizujú svoj stav synchronne na základe danej lokálnej prechodovej funkcie.

Celulárny automat *A* je štvorica

$$A = (S, G, d, f), \tag{1}$$

kde *S* je konečná množina stavov, *G* okolie buniek, $d \in Z^+$ je dimenzia automatu a *f* je lokálna prechodová funkcia.

Ak je daná pozícia bunky ako $\boldsymbol{i}$, $\boldsymbol{i} \in Z^d$, v pravidelnej *d*-dimenzionálnej mriežke, kde $\boldsymbol{i}$ je vektor kladných celých čísel v *d*-dimenzionálnom priestore, je okolie bunky *G* definované takto:

$$G_i = \{\boldsymbol{i}, \boldsymbol{i} + \boldsymbol{r}_1, \boldsymbol{i} + \boldsymbol{r}_2, \ldots, \boldsymbol{i} + \boldsymbol{r}_n\}, \quad (2)$$

kde *n* je veľkosť okolia bunky a $\boldsymbol{r}_j$ je vektor v *d*-dimenzionálnom priestore.

Lokálna prechodová funkcia $f: S^n \rightarrow S$ mapuje stav $s_i \in S$ danej bunky $\boldsymbol{i}$ do iného stavu z množiny *S* na základe funkcie stavov buniek v okolí bunky G_i.

Pre celulárny automat konečnej veľkosti *N* je jeho konfigurácia v čase *t* definovaná ako

$$C(t) = (s_0(t), s_1(t), \ldots, s_{N-1}(t)), \quad (3)$$

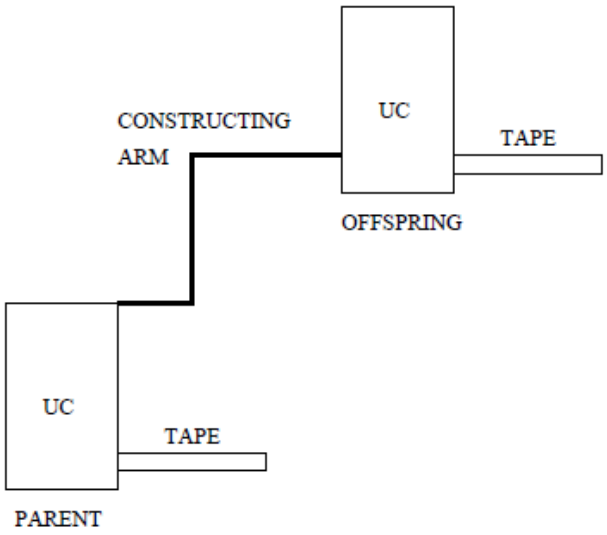
kde $s_i(t) \in S$ je stav bunky $\boldsymbol{i}$ v čase *t*. Vývoj celulárneho automatu v čase je daný iteráciou *globálneho mapovania F*

$$F: C(t-1) \rightarrow C(t), t = 1, 2, 3 \ldots \quad (4)$$

paralelnej aplikácie lokálnej prechodovej funkcie *f* na všetky bunky

2.2 Celulárny automat ako výpočtový model

Aplikácia celulárnych automatov má široký záber vďaka jeho výpočtovým vlastnostiam, ktorými sa zaoberal jeho autor von Neumann [vN66], konkrétne sa snažil zistiť, či je možné navrhnúť automat, ktorý je schopný sebareplikácie. Navrhol *2*-rozmerný 29 stavový automat využívajúci 5-okolie, v ktorom môže byť obsiahnutý univerzálny počítač, ktorý je svojou výpočtovou silou ekvivalentný Touringovmu stroju [HU79]. Tiež ukázal, ako je možné zostrojiť univerzálny konštruktor, ktorý dokáže skonštruovať ľubovoľnú konfiguráciu, ktorú má uloženú na svojej páske pomocou konštrukčného ramena [ST98]. Ak má na páske svoj popis, dokáže vytvoriť kópiu samého seba. Schematický diagram univerzálneho konštruktora je zobrazený na obrázku 5.



Obrázok 5: *Schematický diagram univerzálneho konštruktora (UC) [Sip97b].*

Sebareplikácia je populárnou oblasťou štúdia celulárnych automatov. Jej cieľom je navrhnúť automat tak, aby dokázal vytvárať kópie elementov nachádzajúcich sa vo vnútri automatu. Langton

vytvoril sebareplikačnú slučku v 2-rozmernom automate pri použití 8 stavov [Lan84]. Replikačná štruktúra pozostáva zo slučky a replikátora, ktorý vytvorí kópiu vedľa originálnej slučky. Vytváranie nových slučiek pokračuje, až kým nie je výpočet automatu ukončený.

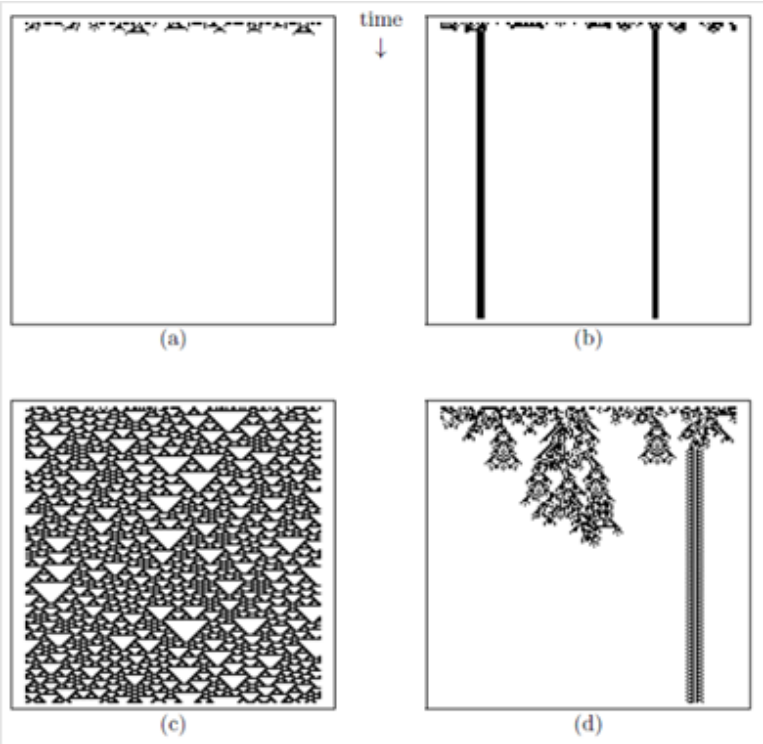
Asi najznámejšou aplikáciou celulárnych automatov je hra *Life*. V 60. rokoch ju vymyslel Conway a ukázal, že je výpočtovo univerzálna [BCG82]. Hra je definovaná takto: každá bunka v 2-rozmernom automate je buď živá (v stave 1) alebo mŕtva (v stave 0) a jej ďalší stav závisí od okolitých buniek na základe štyroch jednoduchých pravidiel [Gar70]:

- Každá živá bunka s jedným alebo žiadnym susedom umiera.
- Každá živá bunka s dvomi alebo tromi susedmi prežíva.
- Každá živá bunka s viac ako tromi susedmi umiera.
- V každej neobývanej bunke s práve tromi živými susedmi vznikne nová živá bunka.

Ďalšie výskumy ukazujú, že celulárne automaty sú vhodné na modelovanie fyzikálnych zákonov priamo v automatoch alebo modelovanie dynamických systémov. Vzťah medzi celulárnymi automatmi a dynamickými systémami skúmal Wolfram [Wol94] a predstavil 4 triedy ich chovania:

1. Trieda I – automat skončí v homogénnej konfigurácii
2. Trieda II – automat skončí v stabilnej nehomogénnej konfigurácii alebo cykle s konečnou periódou
3. Trieda III – automat vytvára náhodné vzory
4. Trieda IV – automat vytvára lokálne nepravidelné štruktúry alebo rozširujúce sa vzory

Príklady tried chovania ukazuje obrázok 6



Obrázok 6: *Ukážka Wolfrámových tried a) Trieda I b) Trieda II c) Trieda III d) Trieda IV [Sip97b].*

2.3 Evolučný návrh celulárnych automatov

Základný model celulárneho automatu prináša niekoľko výhod počnúc jednoduchosťou buniek a končiac masívnym paralelizmom, čo ho pasuje za vhodného kandidáta na hardwarovú implementáciu. Napriek tomu je však samotný návrh celulárneho automatu za účelom špecifickej funkcionality veľmi obtiažnou úlohou, čo predstavuje zásadný problém celulárnych automatov a limituje ich použitie. Riešením tohto problému je automatizácia procesu návrhu a jednou z možností je použitie genetických algoritmov spomínaných v kapitole 3.

Počiatky štúdia využitia genetických algoritmov na evolúciu celulárnych automatov je možné nájsť v [Pac88], na ktoré nadväzuje skupina EVCA (*evolving CA*). Jej výskum [MCH94] spočíva v lepšom pochopení spôsobu akým automaty vykonávajú výpočty, v skúmaní ako najlepšie využiť genetické algoritmy na evolúciu výpočtovo užitočných automatov a v pochopení mechanizmu, pomocou ktorého evolúcia vytvára globálne chovanie systému tvoreného jednoduchými lokálne interagujúcimi časťami.

Pri celulárnych automatoch vykonávajúcich nejaký výpočet predpokladáme, že počiatočná konfigurácia predstavuje zakódovaný vstup výpočtu a výsledok výpočtu je zakódovaný v konfigurácia po *k* krokoch výpočtu automatu. Pomocou genetických algoritmov hľadáme takú sadu pravidiel (lokálnu prechodovú funkciu), aby bol automat schopný vykonať daný výpočet pre každý požadovaný vstup. Vezmime si napríklad problém majority [Sip97a], ktorý určuje, či je v počiatočnej

konfigurácii viac buniek v stave 1 ako v stave 0. Ak áno konfigurácia sa po určitom počte krokov ustáli na homogénnu, kedy sú všetky bunky v stave 1, v opačnom prípade v stave 0. Štandardný genetický algoritmus môže pracovať napríklad s uniformným jednorozmerným automatom o veľkosti 149 buniek s dvomi stavmi a parametrom $r = 3$. To nám dáva $2^{2r+1} = 128$ pravidiel a stavový priestor 2^{128}. Každý kandidátny automat sa spustí s určitou trénovacou množinou počiatočných konfigurácií, ktoré sú náhodne vybrané na základe normálneho rozloženia a ich fitness hodnotu určí na základe správnosti konfigurácie po danom počte krokov. Bolo dokázané [LB95], že pomocou uniformného dvojstavového automatu, nie je možné bezchybne vyriešiť, zatiaľ však nebola nájdená horná hranica najlepšieho riešenia.

2.4 Celulárne programovanie

Výskum celulárnych automatov je väčšinou zameraný na parametre skúmaného výpočtu, ale len málokedy na implementáciu evolučného algoritmu napriek tomu, že jeho výpočet zvykne byť náročný a spotrebuje veľa času najmä kvôli vyhodnocovaniu jednotlivých riešení vo fitness funkcii. Jedným z riešení je paralelizácia [Sip98], ktorá by v princípe nemala spôsobovať problémy, vyžaduje si však rozumnú úpravu existujúceho algoritmu tak, aby splnila obmedzenia daného paralelného systému. Iný prístup navrhuje [Sip98], aby sme sa zamysleli nad možnosťou implementácie v hardwari a položili si otázku, či je v ňom možná evolúcia a predstavuje túto myšlienku na báze neuniformných celulárnych automatov.

Namiesto evolúcie populácie uniformných celulárnych automatov, celulárne programovanie počíta s jedným neuniformným o veľkosti N s náhodne generovanými pravidlami. Fitness každej bunky je akumulovaná z behov automatu s C náhodne vygenerovanými konfiguráciami tak, že ak je bunka po M behoch automatu v správnom stave pripočíta sa k jej fitness 1, inak 0. Evolúcia pravidiel je vykonaná vždy po týchto C behoch v zmysle lokálnej interakcie, kde sú genetické operátory aplikované len medzi bunkami a riadi sa počtom susediacich buniek s lepšou fitness ako aktuálna bunka. Pseudokód algoritmu je možné nájsť v [Sip98]:

Dva hlavné rozdiely medzi celulárnym programovaním a genetickým algoritmom sú v tom, že genetický návrh počíta s populáciou automatov, ktoré sú ohodnotené na základe fitness a používa genetický operátor kríženia medzi dvoma jedincami tejto populácie s ohľadom na ich fitness hodnoty, teda automaty sa vyvíjajú globálne, napriek tomu, že ich výpočet prebieha lokálne. Celulárne programovanie vykonáva oba tieto procesy lokálne a fitness prideľuje jednotlivým bunkám nie automatu ako celku. Evolučný návrh navyše pracuje s nezávislými riešeniami zakódovanými v populácii, kým celulárne programovanie *koevolvuje*, čo znamená, že fitness bunky závisí na jej susedoch.

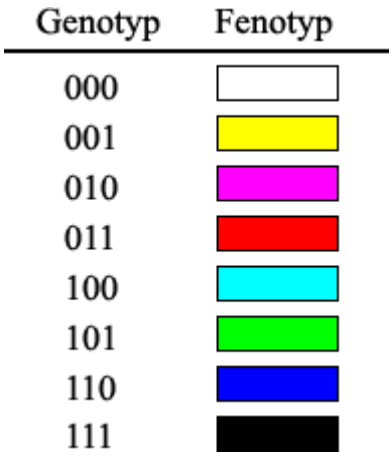
3 Genetické algoritmy

Genetické algoritmy sú počítačové modely založené na genetike a evolúcii v biológii [Mun08]. Patria medzi základné stochastické prehľadávacie algoritmy [KPT00] a vylepšujú algoritmy slepého prehľadávania tak, že usmerňujú náhodné generovanie bodov k hodnotám poskytujúcim kvalitnejšie riešenie.

Celá myšlienka by nemohla vzniknúť bez Darwinovej teórie evolúcie o pôvode druhov [Dar06], ktorá spočíva v prirodzenom výbere a prežití tých najlepších jedincov druhu. Ako je možné vidieť, tento princíp funguje veľmi dobre v prírode a inšpiruje k jeho simulácii a využitiu na riešenie problémov z rôznych oblastí. Ako prvý zhmotnil túto myšlienku v roku 1975 John Holland a vytvoril genetický algoritmus [SD08].

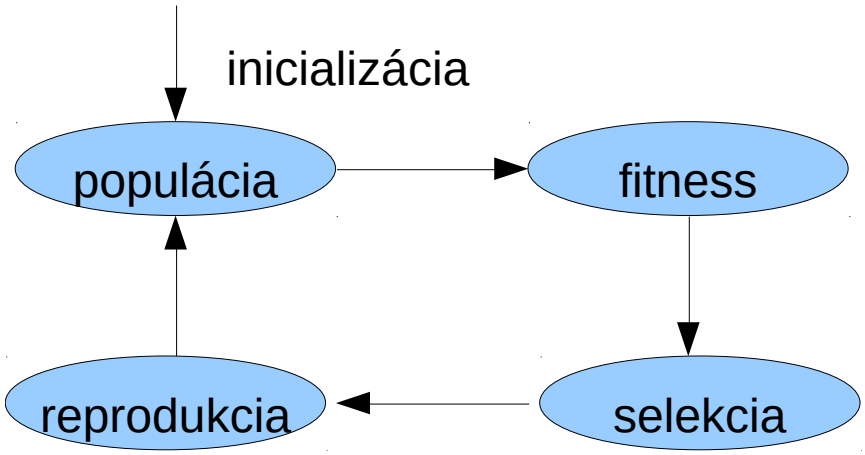
3.1 Základné pojmy a popis algoritmu

Genetický algoritmus pracuje analogicky k procesom v prírode nad danou populáciou jedincov. Keďže aj v prírode evolúcia operuje skôr s chromozómami ako s organizmami [SD08], môžeme si pod jedincom v populácii predstaviť práve chromozóm, ktorý môže byť reprezentovaný binárnym vektorom pevnej dĺžky. Populácia je potom množina takýchto binárnych vektorov. Každý bit chromozómu môže kódovať nejakú vlastnosť organizmu, bit nastavený na hodnotu 1 znamená, že táto vlastnosť je v organizme prítomná, hodnota 0 naopak indikuje neprítomnosť danej vlastnosti. Súbor zakódovaných vlastností v chromozóme sa nazýva genotyp a vyjadrením týchto vlastností dostávame fenotyp, čo predstavuje kandidátne riešenie daného problému. Majme napríklad kvet, ktorého farba je zakódovaná pomocou troch základných farieb azúrovou, purpurovou a žltou (CMY) a konkrétny chromozóm 101. Ten je genotypom pre fenotyp zelenej farby kvetu. Kódovanie farieb v chromozóme formátu CMY ukazuje obrázok 7.



Obrázok 7: *Zobrazenie genotypu na fenotyp pre kódovanie farieb kvetu vo formáte CMY.*

Na začiatku výpočtu je väčšinou náhodne vygenerovaná prvá populácia, niekedy sa používajú heuristiky, ktorými sa dosiahne „lepšia“ počiatočná generácia, a tým sa urýchli čas, ktorý vedie k nájdeniu chromozómu reprezentujúceho uspokojivé riešenie. V ďalšom kroku je treba určiť, ktorý jedinci majú najlepšie vlastnosti a sú potenciálnymi kandidátmi na reprodukciu. Slúži k tomu takzvaná fitness funkcia, ktorá sa interpretuje ako zobrazenie chromozómov na kladné reálne čísla [KPT00]. Nasleduje reprodukcia vhodne vybraných chromozómov (chromozómy s vyššou hodnotou fitness majú väčšiu šancu), ktorá spočíva v použití operátorov kríženia a mutácie, ktoré budú podrobnejšie vysvetlené neskôr. Reprodukciou vznikne nová generácia chromozómov rovnakého počtu ako počiatočná generácia. V tomto bode evolúcie je možné niekoľkými spôsobmi vybrať jedincov do novej populácie a proces môže pokračovať ďalej, až kým nie sú splnené terminačné podmienky. Jednotlivé kroky genetického algoritmu sú ilustrované na obrázku 8.



Obrázok 8: *Kroky genetického algoritmu. [KPT00]*

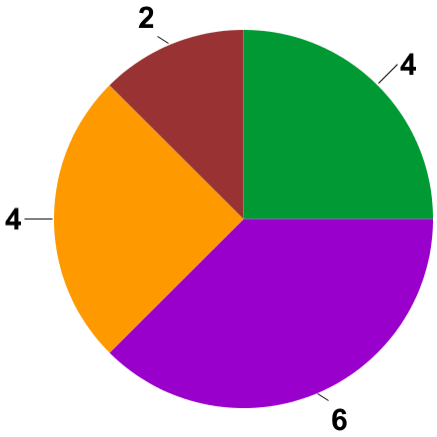
3.2 Výber chromozómov (selekcia)

Výber chromozómov z populácie za účelom reprodukcie je prvý kľúčový krok genetického algoritmu. Proces sa snaží napodobniť Darwinovskú teóriu prirodzeného výberu, kde má silnejší jedinec väčšiu šancu, že bude vybraný, čo v tomto prípade znamená výber jedinca s vyšším ohodnotením.

Metód selekcie existuje hneď niekoľko a medzi najčastejšie používané patrí ruletový mechanizmus. Ruletové koleso vytvoríme tak, že sčítame hodnoty fitness všetkých jedincov populácie a proporcionálne rozdelíme koleso na základe fitness hodnoty jednotlivcov. Teda, čím väčšia je fitness hodnota, tým väčšiu výseč z kolesa jedinec získa a pri roztočení kolesa je väčšia pravdepodobnosť, že „gulička“ zastane práve v tejto výseči. Rozdelenie ruletového kolesa ukazuje

obrázok 9.Formálne [Hyn08]: Uvažujme populáciu o veľkosti $N > 0$ a $f_i \geq 0$ $(i = 1, \ldots, N)$ ako ohodnotenie i-teho jedinca, potom pravdepodobnosť s akou bude jedinec vybratý je daná nasledujúcim vzťahom:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i} \in \{1, \ldots, N\} \tag{5}$$

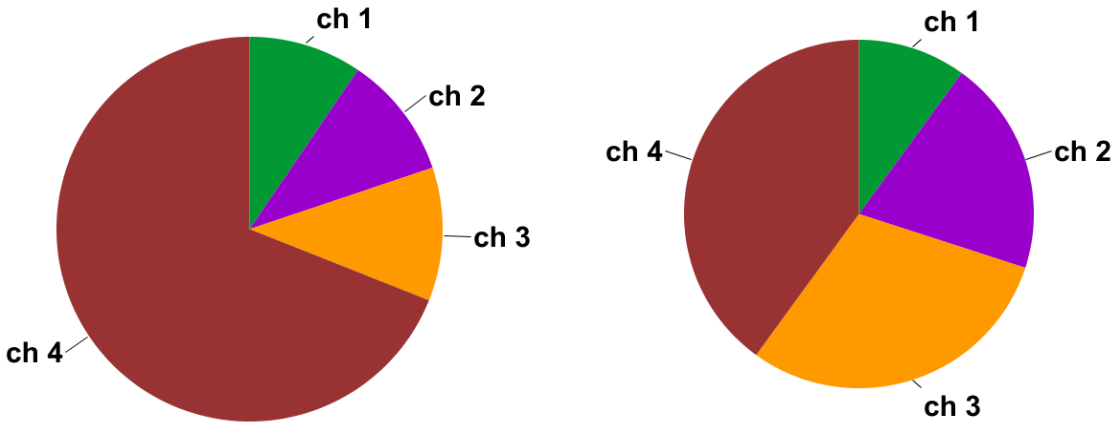


Obrázok 9: *Rozdelenie ruletového kolesa pre hodnoty fitness 6, 4, 4, 2.*

Mechanizmus rulety má nevýhodu v tom, že vytvára vysoký selekčný tlak a silno favorizuje jedincov s vysokou hodnotou fitness, čo môže viesť k rýchlej strate rozmanitosti populácie a predčasnej konvergencii k lokálnemu extrému, a teda k nájdeniu len menej kvalitných riešení riešeniu. Na potlačenie vplyvu nadpriemerných jedincov existuje modifikácia ruletového mechanizmu nazývaná *rank selection* – výber podľa poradia. Jedinci sú usporiadaný podľa fitness hodnoty a ruletové koleso sa rozdelí proporcionálne podľa poradia. Pravdepodobnosť, že bude jedinec vybratý je potom daná vzťahom [Hyn08]:

$$p_i = \frac{i}{\sum_{j=1}^{N} j} \text{ kde N je veľkosť populácie a } i \in \{1, \ldots, N\} \tag{6}$$

Na obrázku 10 je porovnanie ruletového mechanizmu a výberu na základe poradia.



Obrázok 10: *Porovnanie ruletového mechanizmu (vľavo) a výberu na základe poradia (vpravo) pri rovnakých fitness hodnotách jedincov: ch1=1.1 ch2=1.2 ch3=1.3 ch4=8.0.*

Výhodou výberu podľa poradia je nielen to, že potláča nadpriemerne ohodnotených jedincov, ale zároveň udržuje stály selekčný tlak počas celej doby výpočtu [Hyn08]. Na druhej strane pri populácii s malým rozptylom fitness hodnôt môže vzniknúť situácia, kedy sa najlepší a najhorší jedinec líšia len nepatrne, ale pravdepodobnosť, že bude vybratý najlepší jedinec je N-krát väčšia ako pravdepodobnosť, že bude vybratý najhorší jedinec, pričom N je veľkosť populácie.

Posledná prezentovaná metóda selekcie bola inšpirovaná u živočíšnych druhov, kde samce medzi sebou zvádzajú súboj o samičku, tzv. turnajový výber (*tournament selection*). V prípade tejto metódy sa vždy vyberie náhodná skupina dvoch alebo viacerých jedincov, pričom reprodukčného procesu sa ďalej zúčastní ten s najlepšou fitness hodnotou. Pri skupine o dvoch jedincoch je možné metódu vylepšiť tak, že silnejší jedinec má napr. len 95% šancu vyhrať.

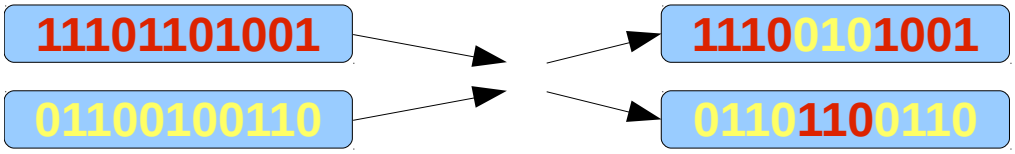
Jednotlivé metódy boli podrobne skúmané a porovnávané, čo viedlo k zisteniu, že pri voľbe vhodných parametrov je pri všetkých metódach dosiahnuť porovnateľných výsledkov a nie je možné vyhlásiť nejakú metódu za lepšiu od ostaných [Hyn08].

3.3 Kríženie a mutácia (reprodukcia)

Pri reprodukcii je možné použiť rôzne genetické operátory. Medzi najznámejšie a najpoužívanejšie patria kríženie a mutácia.

Kríženie využíva dva operandy, teda dva chromozómy a jeho výsledkom sú dva nové chromozómy. Existuje niekoľko variantov kríženia, základom je 1- a viacbodové kríženie využívané najmä pri binárne zakódovaných chromozómoch. Pri 1-bodovom krížení sa náhodne vyberie bod na vstupných chromozómoch. Prvý novovzniknutý chromozóm vznikne konkatenáciou časti prvého pôvodného chromozómu pred týmto bodom a časti druhého chromozómu za týmto bodom. Druhý novovzniknutý chromozóm má naopak prvú časť z druhého pôvodného chromozómu a druhú časť z prvého pôvodného chromozómu. Pri použití viacbodového kríženia sa ďalej striedajú časti z prvého

a druhého pôvodného chromozómu. Hodnota pravdepodobnosti kríženia býva zvolená najčastejšie v rozmedzí 0,6 – 0,9. Na obrázku 11 je zobrazený príklad kríženia.



Obrázok 11: *Ukážka 2-bodového kríženia.*

Kódovania chromozómov reálnymi číslami poskytuje ďalšie možnosti kríženia ako napríklad použitie aritmetických operácií, priemerovania rodičovských hodnôt génov, náhodným generovaním novej hodnoty, atď.

Operátor mutácie by sa mal vykonať s oveľa menšou pravdepodobnosťou ako kríženie, pretože vnáša do chromozómu úplne novú informáciu, ktorá môže výrazne zvýšiť, ale aj znížiť celkovú fitness hodnotu chromozómu a tým napomáha evolúcii dostať sa z lokálneho extrému. Vstupom je len jeden chromozóm, na ktorom sa náhodne vyberie jeden gén, ktorého hodnota zmutuje. V binárnom kódovaní sa jedná o znegovanie náhodne vybraného bitu pri reálnych číslach je možností opäť viac, najčastejšie sa používa náhodná hodnota. Príklad mutácie v binárnom kódovaní uvádza obrázok 12.

Obrázok 12: *Ukážka mutácie.*

3.4 Nová populácia

Tvorba novej populácie značne závisí od použitia genetického algoritmu. Najjednoduchším prípadom je nová populácia, ktorá sa skladá výlučne z nových jedincov a stará populácia vymrie úplne. Je teda veľmi malá pravdepodobnosť, že sľubne vyzerajúci jedinec prejde procesom selekcie a zároveň ho neovplyvnia genetické operátory a prenesie svoj genotyp bez straty informácie do ďalšej generácie.

Túto slabinu je možné prekonať hneď niekoľkými spôsobmi. Prvý sa nazýva elitizmus, kedy sa zo starej generácie najlepší jedinec alebo malá skupina najlepších jedincov automaticky dostane do novej generácie. Druhým je takzvaný *steady-state reproduction* [Hyn08], ktorý najskôr zlúči novú a starú generáciu a následne vytvorí novú populáciu z N najlepších jedincov. Výhodou je jeho rýchlejšia konvergencia, nevýhodou je možnosť ľahšieho uviaznutia v lokálnom extréme.

V niektorých prípadoch je dôležité zachovať rôznorodosť populácie a rovnaké či dokonca podobné chromozómy nie sú v populácii žiaduce. Takéto prípady sa dajú riešiť vytesňovacou

stratégiou, kedy potomok súťaží o miesto v populácií s jedným z rodičov v turnaji alebo penalizáciou za prítomnosť podobných jedincov v populácii [Hyn08].

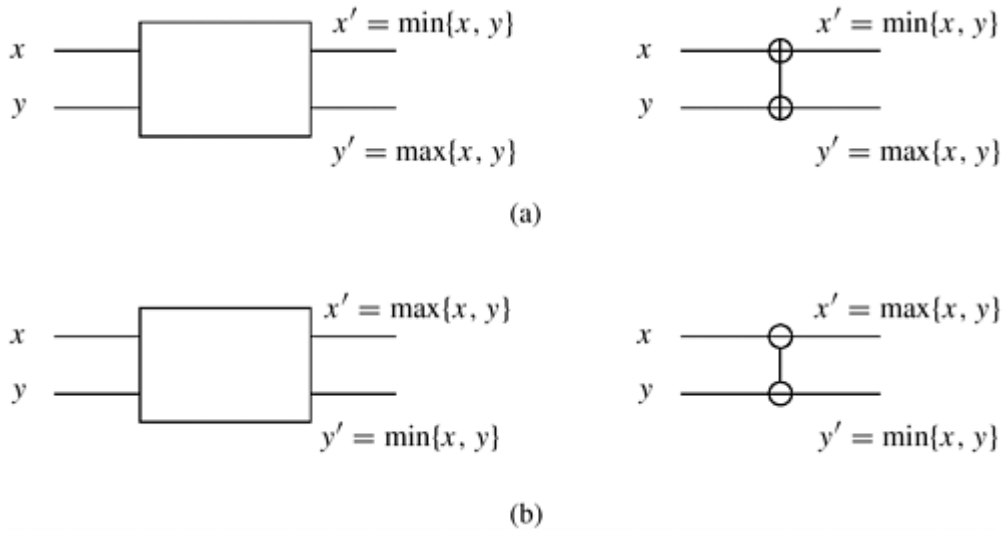
3.5 Evolučný návrh s developmentom

Jedným z problémov evolučného návrhu je škálovateľnosť. So zvyšovaní parametrov daného problému (napr. počet vstupov obvodu) rastie aj priestor, ktorý musí genetický algoritmus prehľadať ako aj počet testovacích vektorov, ktoré musí aplikovať na výpočet fitness funkcie a zistenie správnosti fungovania obvodu.

Existuje niekoľko metód, ktoré prekonávajú tento nedostatok, pre túto prácu bol inšpiráciou embryonálny prístup [Bid04], kedy vývoj hľadaného systému vychádza z nejakého dostatočne jednoduchého počiatočného riešenia - embrya a evolúcia sa snaží nájsť predpis na jeho rozšírenie do finálnej podoby.

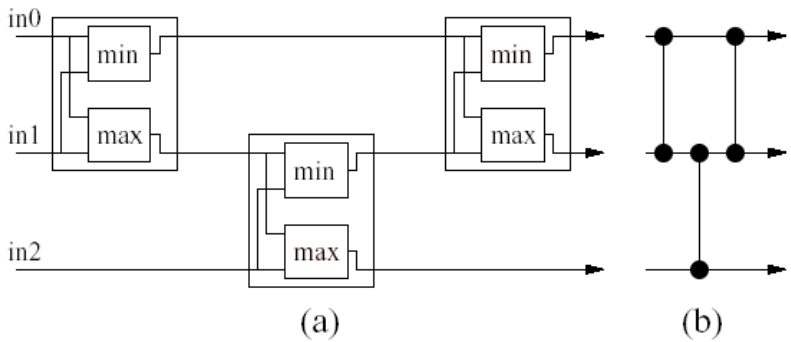
4 Radiace siete

Model radiacich sietí bol predstavený v roku 1954 a jeho história je popísaná v [Knu73]. Radiaca sieť sa skladá z vodičov a porovnávacích modulov nazývaných komparátory. Komparátor je súčiastka s dvomi vstupmi *x, y* a dvomi výstupmi *x', y'*, ktorá porovnáva hodnoty na svojich vstupoch *x* a *y* a zoraďuje tieto hodnoty buď vzostupne alebo zostupne. Pre jednoduchosť budú v texte použité len komparátory, ktoré zoraďujú výstup vzostupne, ak to nebude uvedené inak. Schematicky sú oba typy komparátorov zobrazené na obrázku 13.



Obrázok 13:*Schématické zobrazenie komparátora, ktorý zoraďuje výstup a) vzostupne, b) zostupne [GGK+03].*

Vodiče prenášajú vstupné hodnoty na výstup a komparátory sú medzi nimi vhodne rozložené tak, aby sme na výstupe dostali zoradené vstupné hodnoty. Hlavnou výhodou radiacich sietí je, že sekvencia komparátorov je dopredu nemenná pre ľubovoľnú vstupnú kombináciu danej dĺžky. To ich predurčuje k paralelizácii a k hardwarovej implementácii. Obrázok ukazuje jednoduchú radiacu sieť obsahujúcu 3 komparátory.



Obrázok 14: *Ukážka 3-vtupovej radiacej siete zobrazenej a) s tromi komparátormi b) pomocou zjednodušeného značenia [Bid04].*

Radiace siete klasifikujeme podľa dvoch parametrov: počet použitých komparátorov a oneskorenie siete, ktorej hodnota závisí od počtu úrovní siete, pričom v jednej úrovní sa nachádzajú len komparátory ktoré sú schopné operovať paralelne, t.j. majú navzájom nezávislé vstupy a výstupy. Výskum zameraný na návrh optimálnych sietí našiel siete optimálnych veľkostí pre počet vstupov $n \leq 8$ [Knu73] a siete s optimálnym oneskorením pre počet vstupov $n \leq 10$ [Par91a]. Pre $n \leq 16$ [Knu73] sú známe najlepšie hodnoty veľkosti a oneskorenia sietí (nie optimálne). Pre $n > 16$ sa pre dosiahnutie vyhovujúcich parametrov využíva Batcherovej [Bat68] *odd-even* radiacej siete, ktorá je konštruovaná rekurzívne a dosahuje oneskorenie $(\log n)(\log n + 1)/2$ pri počte komparátorov $n(\log n)(\log n - 1)/4 + n - 1$ [Par92]. Parametre pre radiace siete o n vstupoch, kde , $n \leq 16$ sú uvedené v tabuľke 2.

Počet vstupov	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Oneskorenie	0	1	3	3	5	5	6	6	7	8	8	9	10	10	10	10
Komparátorov	0	1	3	5	9	12	16	19	25	29	35	39	45	51	56	60

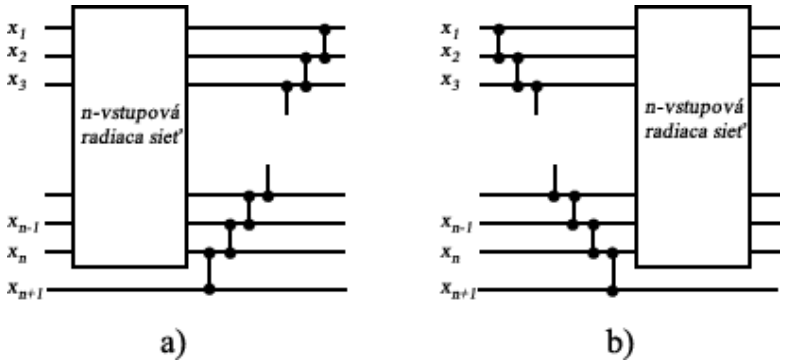
Tabuľka 2: *Optimálne a doposiaľ najlepšie známe hodnoty oneskorení a počtu komparátorov v radiacich sieťach.*

Bolo by užitočné, aby sme vedeli určiť či daná radiaca sieť pracuje správne. Na verifikáciu radiacej siete o veľkosti N potrebujeme otestovať *N!* vstupných kombinácii. Toto číslo však môžeme zredukovať vďaka princípu zvanému *zero-one principle*, ktorý je definovaný nasledovne:

Ak sieť s *n* vstupmi dokáže zoradiť 2^n rôznych postupností jednotiek a núl do neklesajúcej postupnosti, tak dokáže zoradiť ľubovoľnú postupnosť *n* čísel do neklesajúcej postupnosti. Vetu a jej dôkaz je možné nájsť v [Knu73].

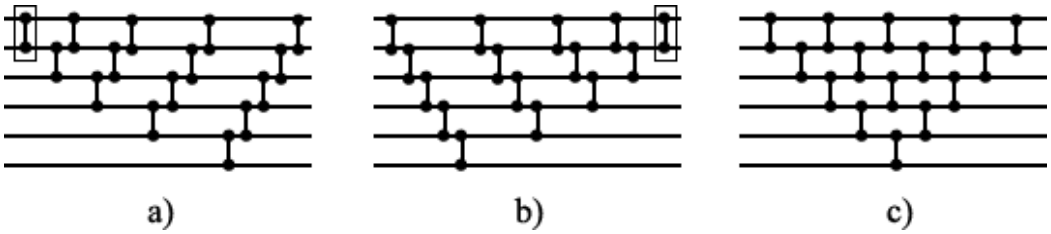
Radiace siete sú väčšinou navrhované pre pevný počet vstupov. Existujú dva jednoduché spôsoby ako skonštruovať radiacu sieť pre *n+1* vstupov ak máme k dispozície *n*-vstupovú sieť, použitím princípu vloženia založeného na algoritme *insertion sort* alebo princípu výberu založeného na algoritme *bubble sort* [Knu73]. Obrázok 15 ukazuje, ako je možné a) (n+1)-vý vstup vložiť na správne miesto potom, čo existujúca sieť usporiadala prvých *n* vstupov a b) ako vybrať najväčšiu hodnotu predtým než pomocou existujúcej siete usporiadame zvyšné vstupy.

Obrázok 16 uvádza príklad opakovaného použitia spomínaných princípov na 2-vstupovú radiacu sieť, v častiach a) a b) vyznačenú obdĺžnikom. Napriek tomu, že komparátory majú v jednotlivých prípadoch zdanlivo odlišnú polohu, pri podrobnejšom skúmaní si môžem všimnúť, že niektoré komparátory sú v oboch prípadoch na tej istej jednej úrovni a budú pracovať paralelne a obe siete sú tým pádom identické.



Obrázok 15: *Tvorba (n+1)-vstupovej radiacej siete z n-vstupovej radiacej siete a) vložením b) výberom [Knu73].*

Je zrejmé, že tieto spôsoby vytvárania ľubovoľne veľkých sietí vedú k hodnotám oveľa väčším ako je optimálny počet komparátorov a oneskorenie siete [SB05]. Vylepšenie týchto základných princípov použitím evolučných techník je možné nájsť v [SB05] a [Bid10].



Obrázok 16: *Príklady sietí vytvorených pomocou princípu a) vloženia b) výberu. Časť c) ukazuje tieto siete pri paralelnom spracovaní komparátorov [Knu73].*

4.1 Formálna definícia

Parberry [Par91b] definuje radiace siete týmto spôsobom:

Nech *C* je sieť komparátorov. Hodnotu na vodiči i a úrovni j pre vstupy $x = (x_1, \ldots, x_n)$ zapisujeme ako $V(C, x, i, j)$. $V(C, x, i, 0) = x_i$ a pre $j > 0$ plati:

- Ak existuje komparátor medzi vodičom i a $k > i$ na úrovni j, potom
$$V(C, x, i, j) = \min(V(C, x, i, j-1), V(C, x, k, j-1)). \quad (7)$$
- Ak existuje komparátor medzi vodičom i a $k < i$ na úrovni j, potom
$$V(C, x, i, j) = \max(V(C, x, i, j-1), V(C, x, k, j-1)). \quad (8)$$
- Inak $V(C, x, i, j) = V(C, x, i, j-1)$.

Výstup *n*-vstupovej, *d*-úrovňovej siete komparátorov *C* pre vstup *x* je
$$V(C, x) = (V(C, x, 1, d), V(C, x, 2, d), \ldots, V(C, x, n, d)). \quad (9)$$
Ak pre všetky vstupy x je $V(C, x)$ neklesajúca postupnosť, potom je *C* radiaca sieť.

4.2 Návrh radiacich sietí bežnými metódami

Návrh radiacej siete spočíva v riešení radiaceho problému (*a sorting problem*), teda v nájdení takej postupnosti operácií porovnania a výmeny (*compare and swap*), ktorá zoradí *n* prvkov do neklesajúcej postupnosti. Na rozdiel od niektorých typov radenia, akým je napríklad *mergesort*, musí byť táto postupnosť komparátorov nemenná pre akýkoľvek dátový súbor na vstupe [Par91a]. Radiacim problémom sa zaoberali už v počiatkoch paralelného počítanie Bose a Nelson [BN62] a na ich prácu nadviazali Floyd a Knuth [FK73].

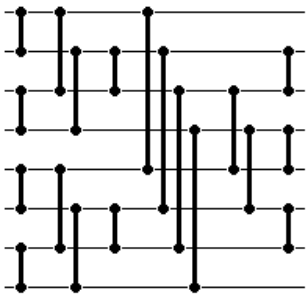
Nemennú postupnosť komparátorov tvorí napríklad *bubblesort,* jeho nevýhodou je však veľké oneskorenie a vysoký počet komparátorov, čomu sa snažíme pri návrhu sietí vyvarovať. Počas dlhodobého výskumu sa osvedčilo niekoľko metód, optimálne siete sa však podarilo nájsť len pre nízky počet vstupov ako je uvedené v úvode tejto kapitoly.

4.2.1 Odd-even mergesort

Táto metóda pracuje na princípe tzv. zotriedenia (*merge*), kedy z dvoch alebo viacerých neklesajúcich postupností vytvorí jednu neklesajúcu postupnosť [Bat68].

Na obrázku 17 je zobrazená časť radiacej siete, ktorá zotriedi dve neklesajúce postupnosti a_1, a_2, ..., a_n a b_1, b_2, ..., b_n do výslednej neklesajúcej postupnosti c_{2n}. Zotriedenie dvoch jednoprvkových postupností dosiahneme ich porovnaním. Na zotriedenie dvoch *n*-prvkových postupnosti potrebujeme vytvoriť dve pomocné siete, do prvej vstupujú prvky z nepárnym indexom (*odd merge*), do druhej prvky s párnym indexom (*even merge*) a výstup týchto sietí porovnáme nasledovne:

1. Najmenší prvok z nepárnych indexov ostane na svojom mieste a vo výslednej postupnosti bude prvým prvkom.
2. *i*-ty výstup z párnych indexov porovnáme s *i+1*-tým prvkom z nepárnych indexov a dostaneme *2i*-ty a *2i+1*-tý prvok výslednej postupnosti
3. Najväčší prvok z párnych indexov ostáva na svojom mieste a vo výslednej postupnosti bude posledným prvkom.



Obrázok 17: *Ukážka radiacej siete vytvorenej pomocou algoritmu odd-even mergesort.*

4.2.2 Bitonic sort

Ďalším prezentovaným algoritmom založeným na zotrieďovaní je *bitonic sort*. Oproti predchádzajúcemu algoritmu potrebuje síce väčší počet porovnaní, jeho výhodou však je, že veľkú sieť je možné rozdeliť na niekoľko identických modulov. Ďalším rozdielom je, že nepracuje s dvomi neklesajúcimi sekvenciami, ale jednou bitonickou. Sekvenciu nazývame bitonickou, ak sa táto sekvencia skladá z dvoch spojených sekvencií, z ktorých je jedna neklesajúca a druhá nerastúca alebo takúto sekvenciu dostaneme cyklickým posunom jej položiek [GGK+03].

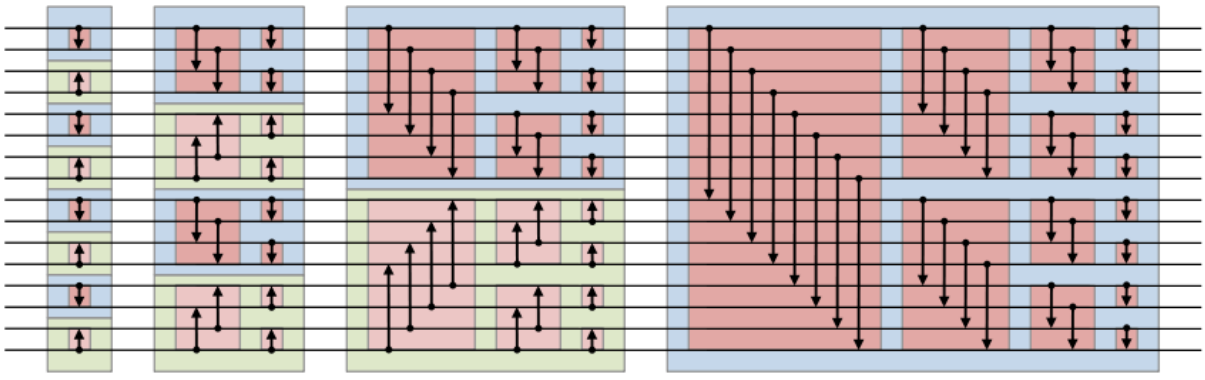
Špeciálnou vlastnosťou bitonickej *2n* prvkovej sekvencie je, že ak ju pomocou nasledujúcich rovníc (10, 11) rozdelíme, získame také dve sekvencie, o ktorých môžeme povedať, že sú bitonické a zároveň žiadny prvok z prvej sekvencie s_1 nie je väčší ako ktorýkoľvek prvok z druhej sekvencie s_2 [Bat68].

$$s_1 = \min(a_1, a_{n+1}), \min(a_2, a_{n+2}), \ldots, \min(a_n, a_{2n}) \quad (10)$$

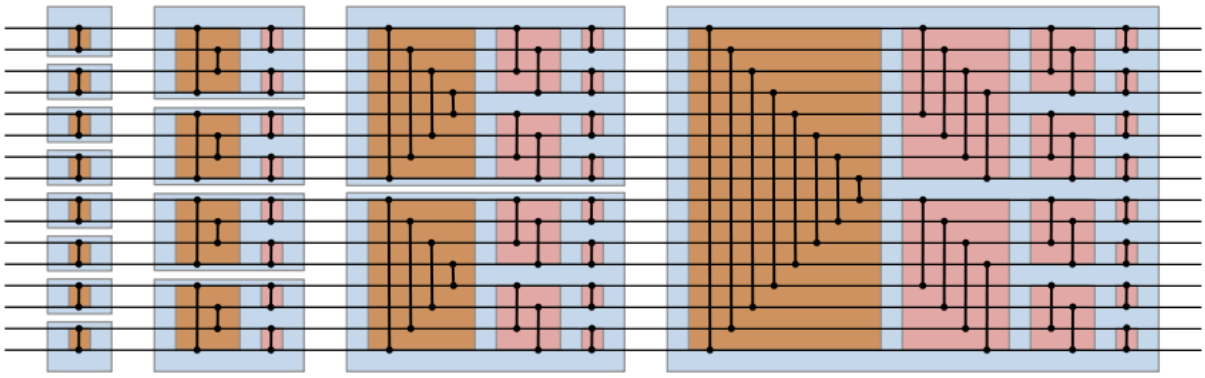
$$s_2 = \max(a_1, a_{n+1}), \max(a_2, a_{n+2}), \ldots, \max(a_n, a_{2n}) \quad (11)$$

Vďaka tejto vlastnosti sa pôvodný problém zoradenia *2n* prvkov zredukoval na zoradenie dvoch bitonických sekvencií o *n* prvkoch a ich následnej konkatenácie. Pomocou aplikácie rovníc (10) a (11) dostanem stále kratšie bitonické podsekvencie. Tento krok rekurzívne opakujeme, až kým nedosiahneme *2n* jednoprvkových podsekvencií, ktorých spojením vznikne výsledná zoradená neklesajúca postupnosť.

Pri konštrukcii radiacej siete pozostávajúcej z komparátorov, ktoré radia výstup vzostupne, je nutné si uvedomiť, že pri prvej aplikácii rovníc pre *n* vstupov pracujeme s dvomi neklesajúcimi sekvenciami a prispôsobiť tomu zapojenie komparátorov. Obrázky 18 a 19 uvádzajú 16-vstupovú bitonickú sieť pri použití rôznych typov komparátorov.



Obrázok 18: *Bitonická radiaca sieť s použitím 2 typov komparátorov, šípky označujú kam smeruje väčší výsledok porovnania.*



Obrázok 19: *Bitonická radiaca sieť s použitím jedného typu komparátora, ktorý zoraďuje výstup vzostupne.*

4.3 Celulárny automat ako generátor radiacich sietí

Väčšina prác zameraných na štruktúrny design uvažuje s reprezentáciou štruktúr priamo v celulárnom automate. V tejto práci však bude použitá modifikácia 1-rozmerného automatu tak, aby bol schopný generovať radiacu sieť mimo svojho priestoru ako samostatnú štruktúru. Tento prístup bol inšpirovaný výskumom popísanom v [BSV10]. Zásadná zmena spočíva v lokálnej prechodovej funkcii, ktorá okrem nasledujúceho stavu bunky obsahuje predpis na vytvorenie komparátora pre radiacu sieť.

Pre *n*-vstupovú radiacu sieť je napríklad možné použiť 1-rozmerný celulárny automat obsahujúci *n* buniek [BVS10]. Každý vstup siete je reprezentovaný jednou bunkou automatu a každá táto bunka generuje jeden komparátor v každom kroku behu automatu. Generovaný komparátor je špecifikovaný ako súčasť pravidla lokálnej prechodovej funkcie. V jednom kroku automatu môže byť vygenerovaných až *n* komparátorov, niektoré však nemusia byť platné pre danú sieť. Aby bol proces generovania deterministický, komparátory sa do siete pridávajú spôsobom navrhovaným v [BVS10]. Každej bunke sa priradí poradie od *0* po *n-1*, pre automat so štyrmi bunkami: $c_0c_1c_2c_3$ sa vygeneruje séria komparátorov $C_{0,0}C_{1,0}C_{2,0}C_{3,0}C_{0,1}C_{1,1}C_{2,1}C_{3,1}C_{0,2}C_{1,2}C_{2,2}C_{3,2}$, kde $C_{i,j}$ reprezentuje komparátor generovaný bunkou c_i v *j*-tom kroku výpočtu.

V nasledujúcich podkapitolách bude popísané aké sú možnosti kódovania generovania komparátorov v lokálnej prechodovej funkcii a ako je možné tieto pravidlá reprezentovať v chromozóme genetického algoritmu.

4.3.1 Možnosti kódovania generovania komparátorov v prechodovej funkcii

Rozšírením pravidiel lokálnej prechodovej funkcie automatu sa naskytá otázka ako v nich vhodne reprezentovať generované komparátory. Spôsobov existuje určite viac, v tomto texte budú prezentované dva, ktoré sú navrhované aj v [BSV10].

Prvou možnosťou je absolútne kódovanie, ktoré kóduje komparátory priamo. Do pravidiel je pridaná dvojica čísel w_1 a w_2, ktoré spĺňajú podmienku $w_1 < w_2$. Upravené pravidlo potom vyzerá nasledovne:

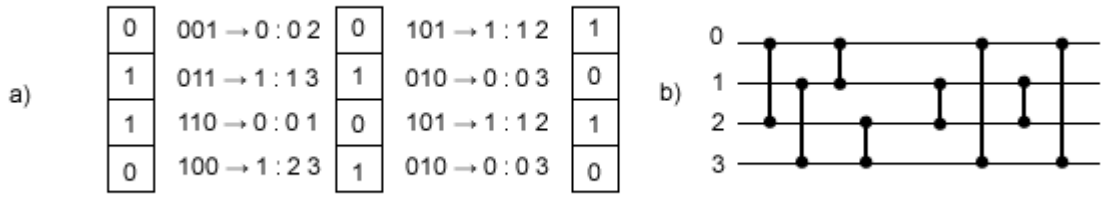
$$c_{i-1}c_ic_{i+1} \rightarrow s_{new} : w_1\ w_2, \tag{12}$$

kde $c_{i-1}c_ic_{i+1}$ popisuje okolie bunky c_i s aktuálnymi stavmi, s_{new} je nový stav bunky c_i a dvojica $w_1\ w_2$ predstavujú vstupy generovaného komparátora. Rozsah hodnôt tejto dvojice je 0 až *n-1*, kde *n* je počet vstupov radiacej siete ako aj počet buniek automatu.

Ako príklad si predstavme nasledovné pravidlá lokálnej prechodovej funkcie celulárneho automatu

000 → 1 : 1 2, 001 → 0 : 0 2, 010 → 0 : 0 3, 011 → 1 : 1 3, 100 → 1 : 2 3, 101 → 1 : 1 2,
110 → 0 : 0 1, 111 → 0 : 1 2

a celulárny automat s počiatočným stavom 0110. Na obrázku 20 sú uvedené postupné konfigurácie celulárneho automatu v čase a ním vygenerované komparátory radiacej siete.



Obrázok 20: *a) Konfigurácie celulárneho automatu a b) ním generovaná radiaca sieť. Posledné tri komparátory sa nikdy nepoužijú a v rámci optimalizácie môžu byť zmazané.*

Druhou možnosťou je relatívne kódovanie, ktoré k zakódovaniu využíva pozíciu bunky. Do pravidiel je pridaná dvojica čísel *r* a *d* a jedno pravidlo po úprave vyzerá takto:

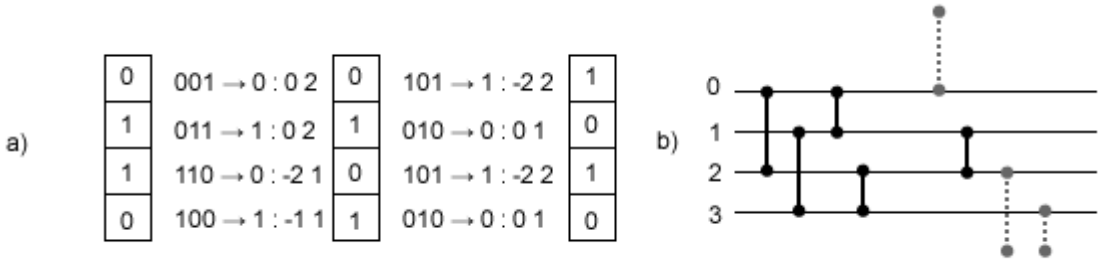
$$c_{i-1}c_ic_{i+1} \rightarrow s_{new} : r\ d, \tag{13}$$

kde význam časti naľavo od dvojbodky ostáva oproti absolútnemu kódovaniu nezmenený a časť napravo má nasledujúci význam. Hodnota *r* špecifikuje vstup w_1 generovaného komparátora relatívne od bunky c_i tak, že $w_1 = i + r$. Hodnota *d* špecifikuje „šírku“ komparátora, teda ako ďaleko sú od seba vzdialené jeho vstupy, w_2 sa dá potom vypočítať ako $w_2 = w_1 + d$. Relatívne kódovanie je možné nastavovať pomocou dvoch parametrov. Parameter *R* určuje rozsah hodnôt *r*, ktoré môže nadobúdať hodnoty od –*R* po *R*. ďalším parametrom je maximálna šírka *D*.

V tomto prípade pozmeníme oproti predchádzajúcemu príkladu s absolútnym kódovaním len časť v pravidlách lokálnej prechodovej funkcie celulárneho automatu, ktorá kóduje generovaný komparátor:

000 → 1 : 0 1, 001 → 0 : 0 2, 010 → 0 : 0 1, 011 → 1 : 0 2, 100 → 1 : -1 1, 101 → 1 : -2 1, 110 → 0 : -2 1, 111 → 0 : -1 2,

počiatočný stav ponecháme 0110. Obrázok 21 uvádza konfigurácie celulárneho automatu a ním vygenerované komparátory radiacej siete. Môžeme si všimnúť, že boli vygenerované komparátory čiastočne mimo radiacej siete. Tieto z výslednej radiacej siete vypustíme.



Obrázok 21: *a) Konfigurácie celulárneho automatu a b) ním generovaná radiaca sieť. Šedé komparátory sú neplatné a nebudú zapísané do výslednej radiacej siete.*

4.3.2 Reprezentácia v chromozóme genetického algoritmu

Pri použití celulárnych automatov by bol opäť konvenčný návrh komplikovaný, preto zapojíme evolučný návrh. Vyžaduje si to malé úpravy v reprezentácii riešení v chromozóme. Prvých *n* génov chromozómu bude kódovať počiatočný stav automatu, ktorý bude tiež podliehať evolúcii. Každá ďalšia trojica bude kódovať nasledujúci stav danej bunky a dva parametre absolútneho alebo relatívneho kódovania. Štruktúra takéhoto chromozómu bude mať formu [BSV10]:

$$s_0 s_1 \dots s_{n-1} ns_0 x_0 y_0 \; ns_1 x_1 y_1 \dots ns_{|Q|^3-1} x_{|Q|^3-1} y_{|Q|^3-1}, \tag{14}$$

kde s_i je počiatočný stav *i*-tej bunky ($i = 0,1, \dots n-1$), ns_j je nový stav pre kombináciu stavov v okolí bunky označený indexom $j = 0,1, \dots, |Q|^3 - 1$, kde $|Q|$ je počet možných stavov bunky, x_j a y_j sú informácie pre vygenerovanie komparátora. Ukážku reprezentácie problému v chromozóme vyobrazuje tabuľka 3.

Kódovanie	**Počiatočný stav CA**	**Pravidlá prechodovej funkcie**							
		000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
absolútne	0110	1 1 2	0 0 2	0 0 3	1 1 3	1 2 3	1 1 2	0 0 1	0 1 2
relatívne	0110	1 0 1	0 0 2	0 0 1	1 0 2	1-1 1	1-2 2	0-2 1	0-1 2

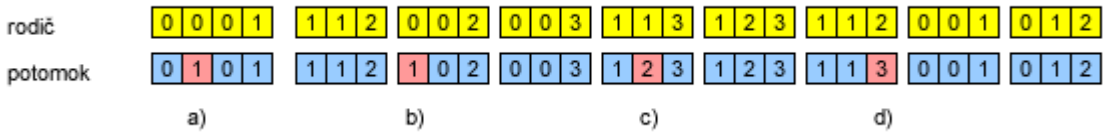
Tabuľka 3*:Ukážka reprezentácie problému v chromozóme za použitia absolútneho a relatívneho kódovania.*

4.3.3 Genetický algoritmus

Kroky jednoduchého genetického algoritmu ostávajú oproti definíciám popísaných v kapitole 2 nezmenené, sú však prispôsobené nášmu problému. Generovanie radiacich sieti v našom prípade pripomína generovanie kombinačných obvodov s použitím kartézskeho genetického programovania, v ktorom sa nepoužíva operátor kríženia, ale iba operátor mutácie. Nové výskumy, napríklad [CVM07] sa snažia rozšíriť kartézske genetické programovanie o operátor kríženia, z toho dôvodu je kríženie popísané v tomto texte ako aj použité pri niektorých experimentoch.

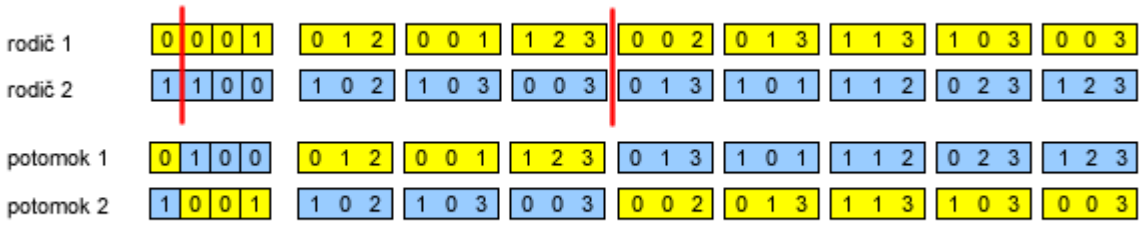
Gény chromozómov populácie inicializujeme na náhodné hodnoty tak, že prvých n génov môže nadobúdať hodnoty od 0 po $|Q|-1$, kde $|Q|$ je počet všetkých možných stavov bunky automatu. Inicializácia trojíc kódujúcich nasledujúci stav a parametre komparátora sú tiež generované náhodne, pri absolútnom kódovaní môžu vstupy nadobúdať hodnoty od 0 po $n-1$, pri relatívnom kódovaní je šírka obmedzená užívateľom a posun intervalom $< -R, R >$, kde $R = |Q| - 1$.

V procese mutácie zmení svoju hodnotu podľa danej pravdepodobnosti práve jeden gén. Keďže pracujeme s nehomogénnymi chromozómami, je potrebné prihliadať na to, v ktorej časti chromozómu sa gén, ktorý mutujeme, nachádza. Pozíciu mutovaného génu určuje náhodne vygenerované číslo z intervalu $< 0, n + |Q|^3 * 3)$, kde otvorený koniec intervalu je dĺžka chromozómu, čiže počet vstupov radiacej siete zvýšený o 3 pre každé pravidlo prechodovej funkcie. Mutovaný gén zvýši svoju hodnotu o konštantu 1 a podľa pozície v chromozóme zistí, či táto hodnota spadá do intervalu možných hodnôt pre danú pozíciu a ak nie, nastaví hodnotu na minimálnu z tohto intervalu. Všetky možnosti mutácie sú ukázané na obrázku 22.



Obrázok 22:*Ukážka mutácie a) počiatočného stavu CA b) nasledujúceho stavu CA c) a d) vstupov generovaného komparátora.*

Pre potreby tejto práce bol navrhnutý nový operátor kríženia, kedy sa samostatne pomocou jednobodového kríženia kríži prvá časť chromozómu, kde je uložený počiatočný stav celulárneho automatu a samostatne druhá časť chromozómu s pravidlami lokálneho prechodovej funkcie celulárneho automatu. V druhej časti chromozómu ďalej musí byť splnená podmienka, že pôvodné pravidlá sú brané ako celok, a teda jedno pravidlo nemôže byť rozdelené medzi dvoch jedincov. Zmeny v rámci jedného pravidla teda zabezpečuje len mutácia, ktorá by mala byť napriek použitiu kríženia dostatočne veľká. Výber jedincov, ktorí sa majú krížiť je vykonaný na princípe rulety. Navrhovaný operátor ilustruje obrázok 23.



Obrázok 23: *Ukážka navrhovaného operátora kríženia.*

Poslednou otázkou ostáva ako ohodnotiť jednotlivé riešenia. Základom pre fitness funkciu je *zero-one principle* popísaný v predchádzajúcej kapitole, pomocou ktorého vieme určiť koľko postupností núl a jednotiek vygenerovaná radiaca sieť správne zoradila. Z toho vyplýva, že správne fungujúca *n*-vstupová radiace sieť bude mať fitness hodnotu práve 2^n. Ak algoritmus nájde sieť s takouto hodnotou môže celý cyklus ukončiť Fitness hodnotu je ďalej možné rozšíriť, keď chceme optimalizovať počet komparátorov a oneskorenie radiacej siete a to tak, že ak nájdeme správne fungujúcu radiacu sieť zvýšime jej hodnotu o opačnú hodnotu počtu komparátorov a opačnú hodnotu oneskorenia. Ak chceme zvýšiť vplyv týchto parametrov radiacej siete, tak pred pričítaním k pôvodnej fitness hodnote, vynásobíme opačné hodnoty vhodnou konštantou, napríklad nejakou mocninou čísla 10. Pre tento spôsob získavania fitness hodnoty už nevieme dopredu zistiť jej maximálnu hodnotu, preto na ukončenie algoritmu, použijeme inú ukončovaciu podmienku, kedy prihliadame na počet generácií, počas ktorého sa maximálna hodnota fitness funkcie nezlepšila.

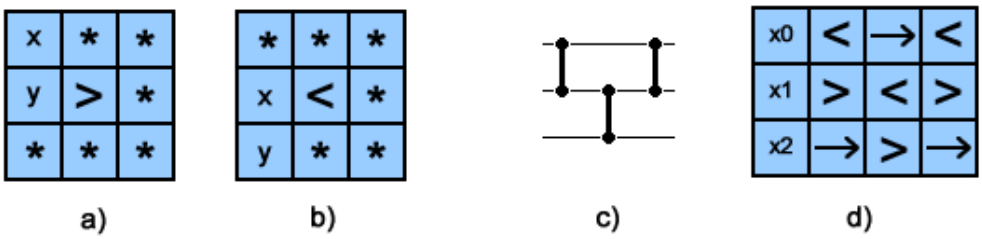
Výslednú radiacu sieť získanú pomocou genetického algoritmu môžeme ďalej optimalizovať tak, že z nej odstránime nepotrebné komparátory. Opäť k tomu poslúži *zero-one principle*, ktorý pri testovaní všetkých možných postupnosti núl a jednotiek označí, koľkokrát bol daný komparátor použitý. Nepoužité komparátory sa zo siete potom jednoducho odstránia.

4.4 Použitie 2D celulárneho automatu na konštrukciu radiacich sietí

K myšlienke priamej reprezentácie radiacej siete v dvojrozmernom celulárnom automate inšpiruje text [Sip97a], kde sú za pomoci jednouchých pravidiel propagácie, operácie XOR a NAND, implementované vodiče, signály a komponenty vykonávajúce logické operácie ako súčasť neuniformného celulárneho automatu.

Radiacu sieť si môžeme predstaviť tak, že jej vstupy budú v prvom stĺpci celulárneho automatu a zoradenú postupnosť budeme očakávať v poslednom stĺpci. K správnemu fungovaniu siete budú

ešte okrem spomínaných pravidiel potrebné bloky reprezentujúce komparátory prezentované na obrázku 24 spolu s ukážkou jednoduchej radiacej siete.



Obrázok 24: *a) a b) bloky reprezentujúce komparátor, ukážka radiacej siete zobrazená c) pomocou komparátorov, d) v celulárnom automate*

Na obrázku 2čd vidíme, že už pre jednoduchú sieť sme potrebovali o jeden stĺpec automatu viac by sme očakávali. Keď sa zamyslíme, že potrebujeme do jedného komparátora zapojiť vstupy, ktoré spolu nesusedia musíme spraviť medzery medzi vstupmi, aby sa hodnoty mohli v rámci automatu presúvať. Tiež musíme zabezpečiť, aby sa hodnoty dostali na vstup komparátora naraz, prípadne, aby sa na vstupe dokázali zosynchronizovať.

Ako vidíme priama reprezentácie v 2D celulárnom automate so sebou prináša rad úskalí, na základe ktorých bol pre túto prácu zvolený prístup z kapitoly 4.3.

5 Využitie celulárnych automatov na tvorbu generických radiacich sietí.

Táto práca sa zaoberá postupne rozšíriteľnými radiacimi sieťami. Jedná sa o už existujúce radiace siete, ktoré chceme rozšíriť o istý počet nových vstupov a na samotné rozšírenie používame opakovanie známych vzorov pre rozšírenie siete o nízky počet vstupov (1, 2, 3, ...). Najjednoduchší vzor rozširuje existujúcu sieť o 1 vstup na princípe *insertion sort*, ukážku je možné vidieť na obrázku 15 v predchádzajúcej kapitole.

Existujúcu radiacu sieť budeme nazývať embryo, ku ktorému sa pri behu celulárneho automatu s pravidlami lokálnej prechodovej funkcie získanými evolúciou postupne pridávajú vygenerované komparátory. Časť siete vytvorená evolúciou sa nazýva vzor a existuje viacero spôsobov ako jeho opakovaním sieť rozšíriť. Vybrané spôsoby ukážeme na príkladoch, kde ako embryo použijeme dvojvstupovú radiacu sieť (na obrázkoch označené žltou farbou), vzor budeme hľadať po pridaní dvoch nových vstupov a ukážeme si tvorbu 6- a 8-vstupovej siete opakovaním tohto vzoru.

5.1 Opakovanie získaného vzoru

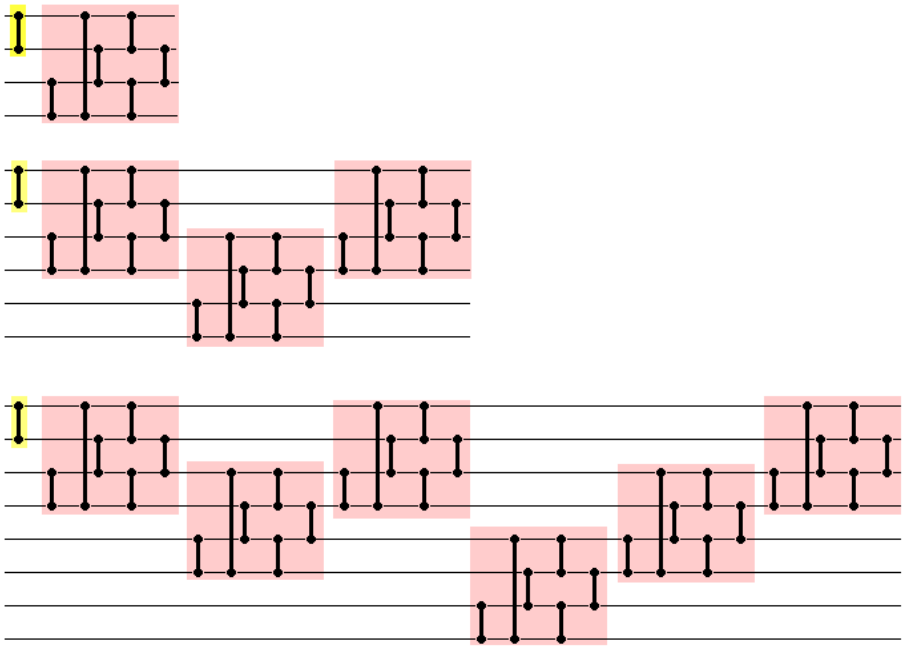
Základný princíp spočíva v opakovaní celého získaného vzoru ako je možné vidieť na obrázku 25. Prvá časť obrázku ukazuje 2-vstupové embryo a získaný vzor. Druhá časť obrázku ukazuje sieť rozšírenú o 2 vstupy. Môžeme si všimnúť, že kópiu vzoru sme preložili sieťou tak, aby pokrývala 2 pridané vstupy a 2 vstupy z pôvodnej siete. Ďalšiu kópiu vzoru umiestnime tak, aby pokrývala 2 vstupy z predchádzajúcej kópie vzoru a 2 vstupy z pôvodnej siete. Tento postup opakujeme, kým nie je pokrytá pôvodná radiaca sieť, čo v našom prípade znamená, že je proces rozširovania ukončený a my sme získali 6 vstupovú radiacu sieť z pôvodnej 4-vstupovej. Pri pridávaní ďalších 2 vstupov označíme aktuálnu 6-vstupovú za pôvodnú a obdobným spôsobom prikladáme kópie vzoru. Postup je formálnejšie popísaný v algoritme 1.

Algoritmus 1: *Rozšírenie radiacej siete opakovaním vzoru*

Vstup: pôvodná sieť, vzor, počet nových vstupov.

Výstup: rozšírená sieť.

1. Vytvor kópiu vzoru.
2. Vypočítaj posun kópie vzoru tak, aby jednou časťou prekrývala nové vstupy alebo poslednú pridanú kópiu vzoru a zvyšnou časťou pôvodnú sieť.
3. Ak nie je pokrytá celá pôvodná sieť, prejdi na bod 1, inak skonči.



Obrázok 25: *Základný princíp tvorenia sietí opakovaním získaného vzoru.*

5.2 Opakovanie vrstiev získaného vzoru

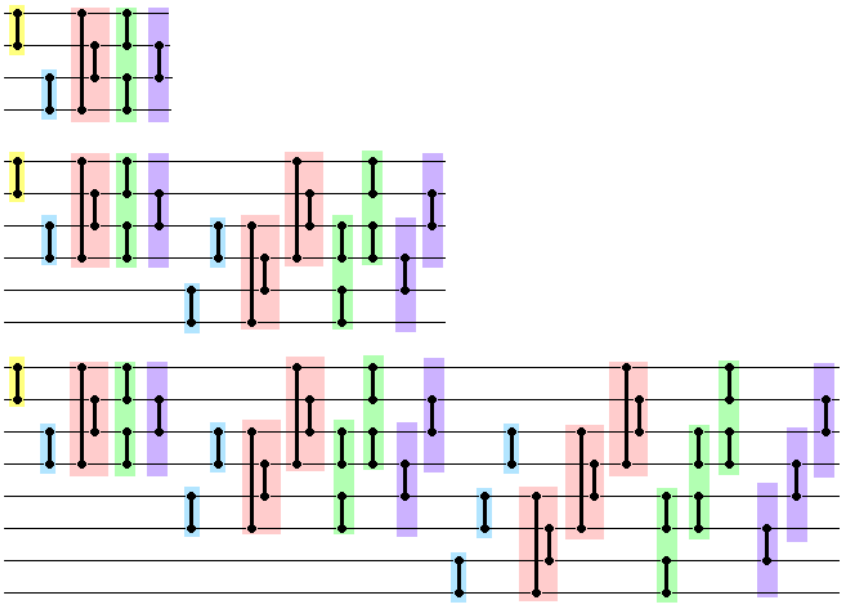
Alternatívny spôsob opakovania vzorov využíva rozdelenie vzoru na vrstvy, pričom každá vrstva obsahuje len komparátory, ktoré sú schopné pracovať paralelne v jednom kroku. Každá vrstva má rovnakú šírku ako pôvodný vzor aj keď komparátory v nej obsiahnuté nevypĺňajú celú šírku vrstvy. Pri pridávaní vrstvy do siete postupujeme obdobne ako pri pridávaní vzoru a kým je to možné pridávame opakovane prvú vrstvu, potom druhú atď. Metódu ilustruje obrázok 26 a popisuje algoritmus 2.

Algoritmus 2: *Rozšírenie radiacej siete opakovaním vzoru*

Vstup: pôvodná sieť, vzor, počet nových vstupov.

Výstup: rozšírená sieť.

1. Rozdeľ vzor do vrstiev, v ktorých komparátory pracujú paralelne a vytvor kópiu prvej vrstvy.
2. Vypočítaj posun kópie vrstvy tak, aby jednou časťou prekrývala nové vstupy alebo poslednú pridanú kópiu vrstvy a zvyšnou časťou pôvodnú sieť.
3. Ak nie je pokrytá celá pôvodná sieť, vytvor kópiu aktuálnej vrstvy a prejdi na bod 1 inak pokračuj bodom 4.
4. Ak je k dispozícii ďalšia vrstva, vytvor jej kópiu a pokračuj bodom 2, inak skonči.

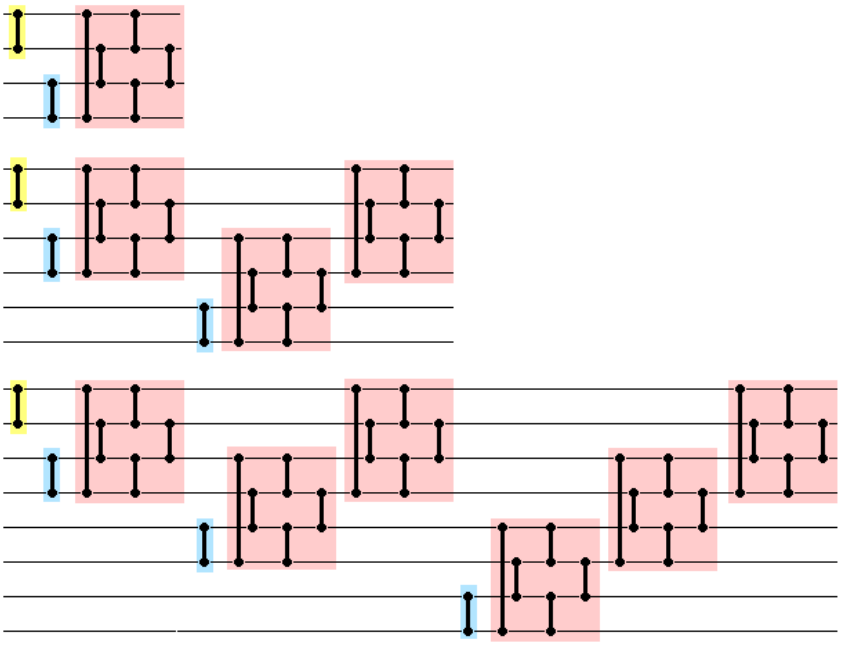


Obrázok 26: *Opakovanie vzorov po vrstvách.*

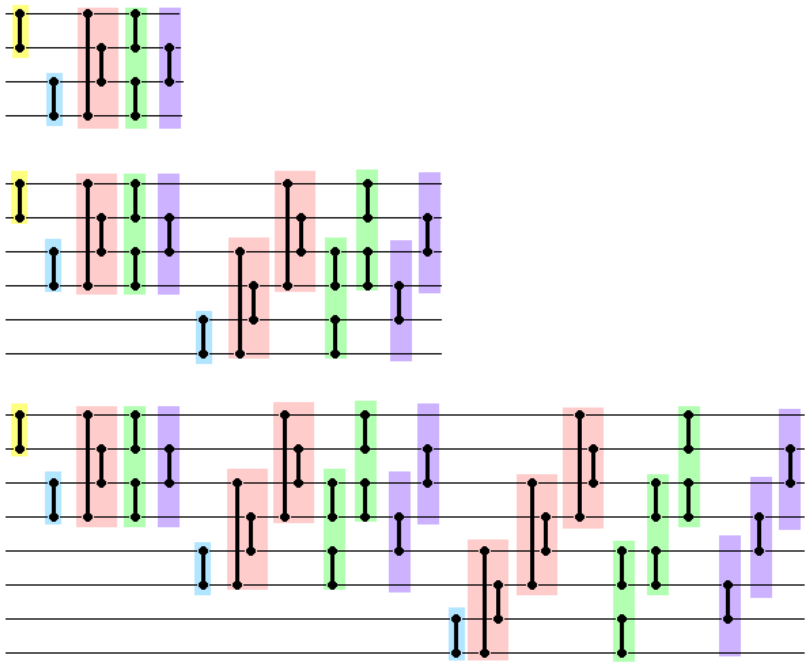
5.3 Vylepšenia a obmedzenia

Možnosť ako sa zbaviť nadbytočných komparátorov je pridávané vstupy najskôr zoradiť a následne aplikovať predchádzajúce metódy ako je možné vidieť na obrázkoch 27 a 28. Do zoradenia môžeme zaradiť aj niekoľko posledných vstupov z embrya, čo môže mať vplyv na parametre siete najmä pri rozšírení o nepárny počet vstupov.

Prezentované spôsoby rozširovania radiacich sietí o nové vstupy majú jedno zásadné obmedzenie, ktorým je počet nových vstupov. Ten nemôže byť väčší ako je šírka embrya. K embryou širokému dva vstupy sa dá teda pridať jeden alebo dva vstupy, 3-vstupové embro sme schopný rozšíriť o jeden, dva alebo tri vstupy, atď. Toto obmedzenie sa dá obísť buď postupným rozširovaním o menší počet vstupov, alebo výmenou embrya a nových vstupov tak, že nové vstupy zoradíme nejakou známou sieťou a rozšírime o počet vstupov pôvodného embrya.



Obrázok 27: *Opakovanie vzorov po zoradení nových vstupov.*



Obrázok 28: *Opakovanie vzorov po vrstvách po zoradení nových vstupov.*

5.4 Vyhodnotenie generickej siete

Na vyhodnotenie správnosti fungovania rozširujúcich sa sietí sa dá použiť *zero-one principle*, ktorý postupne aplikujeme pre každé rozšírenie siete, až kým sieť nedosiahne dostatočnú veľkosť N. V rámci tejto práce to budeme považovať za postačujúce, ale na potvrdenie správnosti fungovania je potrebný formálny dôkaz každého riešenia, čo môže byť inšpiráciou na pokračovanie tejto práce.

6 Experimentálne výsledky

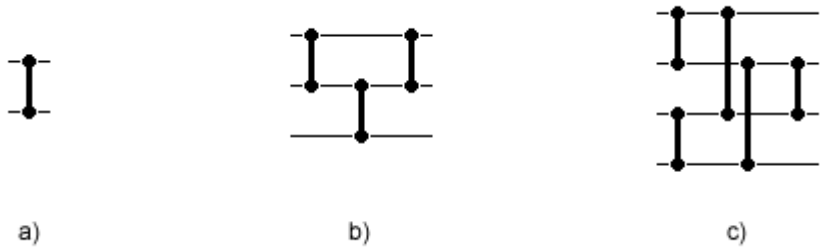
Na použitie techník prezentovaných v kapitole 5 pre tvorbu generických radiacich sietí boli zvolené tri oblasti aplikácie: návrh generických radiacich sietí s párnym počtom vstupov, návrh generických sietí s nepárnym počtom vstupov a návrh generických sietí so striedavo párnym a nepárnym počtom vstupov. Pre každú oblasť bolo vykonaných niekoľko experimentov, výber najlepších z nich bude uvedených v tejto kapitole.

Počiatočné experimenty s nastavením genetického algoritmu viedli k vylúčeniu operátora kríženia, s ktorým neboli dosiahnuté lepšie výsledky ako so samotnou mutáciou. Pravdepodobnosť mutácie sme na základe týchto výsledkov nastavili na 0,8 a túto hodnotu sme používali pri ďalších experimentoch. Populácia bola nastavená na 100 jedincov a evolúcia na 10000 generácií.

Na základe výsledkov bolo pre ďalšie pokračovanie experimentov vybraté absolútne kódovanie z kapitoly 4. Celulárny automat podľa potreby pracoval v 2 až 3 krokoch, pričom jeho bunky sa mohli nachádzať v jednom z 2 až 5 stavov.

6.1 Popis experimentov

V nasledujúcich experimentoch budeme hľadať vzory pre rozšírenie existujúcich radiacich sietí. Jedná sa o hľadanie zotrieďovacích blokov relatívne malej veľkosti. Základnými stavebnými jednotkami sú už spomínané embryá, ku ktorým sa pri behu celulárneho automatu postupne pridávajú nové komparátory. Na účely našich experimentov boli zvolené 3 embryá: 2-, 3- a 4-vstupová radiaca sieť zobrazené na obrázku 29.



Obrázok 29: *Embryá použité na experimenty a) 2-vstupové b) 3-vstupové c) č-4-vstupové.*

Ďalšími stavebnými jednotkami hľadaných vzorov sú radiace bloky, ktoré zoradia nové vstupy predtým, ako ďalej vstúpia do vzoru komparátorov. Prehľad použitých radiacich blokov bude uvedený samostatne v každej oblasti experimentov.

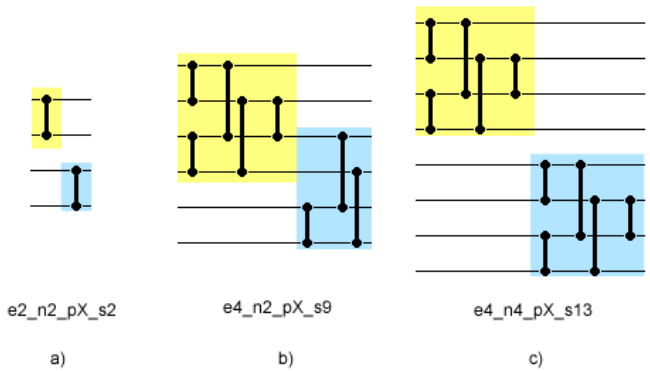
Experimenty sú stručne popísané a doplnené o tabuľku, prípadne obrázok generickej radiacej siete, ktorá tvorí zaujímavý vzor. Na obrázku vzoru je farebne odlíšené embryo, radiaci blok a samotný vzor. Embryo je ohraničené žltou farbou, radiaci blok modrou a vzor ružovou.

V prípadoch, kedy sieť vznikla rozšírením pomocou opakovanie jednotlivých vrstiev vzoru, sú tieto vrstvy odlíšené ďalšími farbami. Informácie o počte komparátorov a oneskorení siete sú pri každom experimente zhrnuté v dvoch tabuľkách s možnosťou porovnania so sieťami vzniknutými konvenčným spôsobom na princípe vkladania ako aj so sieťami vygenerovanými sadou inštrukcií získanou v práci [SB05]. Jednotlivé vzory sú označené reťazcom v tvare „eX_nY_pZ“, kde číslo X udáva počet vstupov embrya, číslo Y počet nových vstupov a číslo Z označuje číslo vzoru a číslo Z číslo vzoru pre dané embryo a nový počet vstupov. Ak bol použitý radiaci blok je k reťazcu ešte pripojené „_sW“, kde W označuje číslo radiaceho bloku. Ak reťazec na konci obsahuje „_L“ sieť, bola vytvorená opakovaním vrstiev vzoru. Najlepšie výsledky sú v danej tabuľke zobrazené kurzívou.

V prvých experimentoch sme rozširovali 2- a 3- vstupové embryá o 1 vstup a snažili sme sa získať vzory, ktoré by generovali siete s obdobnými parametrami ako pri použití princípu vkladania. To sa nám podarilo dosiahnuť práve len pri vzoroch, ktoré generovali rovnaké siete ako spomínaný princíp. Navyše bolo potrebné použitie radiacich blokov, ktoré najskôr zoradia pridaný vstup s istou časťou už existujúcej radiacej siete. Tým pádom ako vzor ostal jediný komparátor, ktorý sa pri rozširovaní posúval a opakoval. Keďže sme v tejto oblasti nedosiahli lepšie ako známe výsledky, v tejto práci ich neuvádzame.

6.2 Návrh generických radiacich sietí s párnym počtom vstupov

V tejto kategórii experimentov boli použité 2-vstupové a 4-vstupové embryá, 2-vstupové embryo bolo rozširované o 2 nové vstupy a 4-vstupové o 2 a 4 vstupy. Zaujímavé výsledky, ktoré budú uvedené v nasledujúcich kapitolách, sme dosiahli pri použití všetkých metód okrem opakovanie vrstiev bez použitia radiaceho bloku. Prehľad použitých radiacich blokov uvádza obrázok 30.



Obrázok 30: *Použité radiace bloky pre jednotlivé embryá a) e2_n2_pX_s2 b) e4_n2_pX_s9 c) e4_n4_pX_s13.*

6.2.1 Experimenty s opakovaním vzorov

Pri použití tejto metódy tvorby generických radiacich sietí sme dosiahli uspokojivé výsledky pri rozširovaní 4-vstupového embrya o 4 vstupy. Vzor *e4_n4_p1* potrebuje menší počet komparátorov počet komparátorov ako konvenčná metóda, nedosahuje však hodnoty získané inštrukciami. Porovnanie uvádza tabuľka 4. Oneskorenie sietí tvorených týmto spôsobom nedosahuje kvality konvenčnej metódy, ako je možné vidieť v tabuľke 5.

	6	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**
konvenčne	15	28	45	66	91	120	153	190	231	276	325	378
inštrukcie	12	22	35	51	70	92	117	145	176	210	247	287
e4_n4_p1		*21*		*53*		*101*		*165*		*245*		*341*
e4_n4_p2		23		59		113		185		275		383
e4_n4_p3		24		62		119		195		290		404

Tabuľka 4: *Počet komparátorov sietí s párnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru.*

	6	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**
konvenčne	9	13	17	21	25	29	33	37	41	45	49	53
inštrukcie	7	11	15	19	23	27	31	35	39	43	47	51
e4_n4_p1		9		23		37		51		65		79
e4_n4_p2		*9*		*21*		*33*		*45*		*57*		*69*
e4_n4_p3		8		21		34		47		60		73

Tabuľka 5: *Oneskorenie sietí s párnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru.*

6.2.2 Experimenty s opakovaním vzorov pri použití radiaci blokov

Hneď prvé experimenty pri zapojení radiaceho bloku do 2-vstupového embrya priniesli kvalitné výsledky a pri opakovaní vzoru *e2_n2_p1_s2* boli dosiahnuté parametre siete zhodné so sieťami generovanými inštrukciami. Výrazné zlepšenie vykazujú všetky vzory získané rozšírením 4-vstupového embrya o 4 vstupy, osobitne potom vzor *e4_n4_p10_s13*, ktorému stačí pre 28 vstupov 224 komparátorov, ktorých zapojenie produkuje oneskorenie s hodnotou 36. Porovnanie počtu komparátorov jednotlivých sietí uvádza tabuľka 6, porovnanie ich oneskorení tabuľka 7.

	6	8	10	12	14	16	18	20	22	24	26	28
konvenčne	15	28	45	66	91	120	153	190	231	276	325	378
inštrukcie	12	22	35	51	70	92	117	145	176	210	247	287
e2_n2_p1_s2	12	22	35	51	70	92	117	145	176	210	247	287
e4_n4_p8_s13		21		48		86		135		195		266
e4_n4_p10_s13		*19*		*42*		*74*		*115*		*165*		*224*
e4_n4_p11_s13		20		45		80		125		180		245

Tabuľka 6: *Počet komparátorov sietí s párnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru pri použití radiaceho bloku.*

	6	8	10	12	14	16	18	20	22	24	26	28
konvenčne	9	13	17	21	25	29	33	37	41	45	49	53
inštrukcie	7	11	15	19	23	27	31	35	39	43	47	51
e2_n2_p1_s2	7	11	15	19	23	27	31	35	39	43	47	51
e4_n4_p8s_13		8		16		24		32		40		48
e4_n4_p10_s13		*6*		*12*		*18*		*24*		*30*		*36*
e4_n4_p11_s13		7		15		23		21		39		47

Tabuľka 7: *Oneskorenie sietí s párnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru pri použití radiaceho bloku.*

6.2.3 Experimenty s opakovaním vrstiev vzorov pri použití radiacich blokov

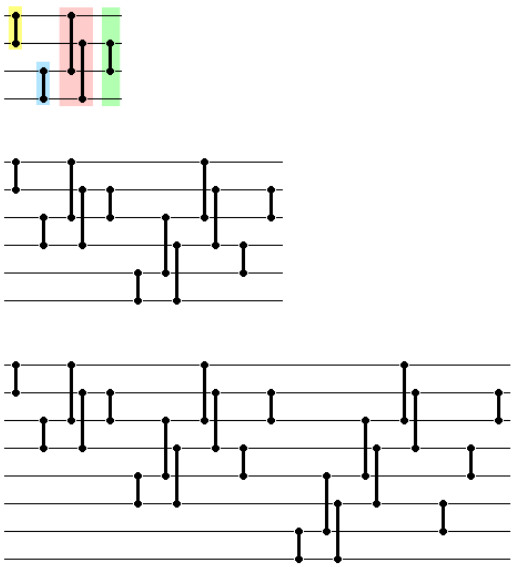
Použitie druhej navrhovanej metódy, pri ktorej sa neopakuje celý vzor ale postupne jeho vrstvy, dokázalo ešte o niečo vylepšiť parametre sietí získaných v podkapitole 6.2.1. Pri porovnaní tabuliek 6 a 8 vidíme, že u rovnakých vzorov nedošlo zmenou metódy opakovanie k žiadnej zmene v počte komparátorov. Oneskorenie novovzniknutých sietí uvádza tabuľka 9. Pomocou vzoru *e2_n2_p1_s2_L* sa podarilo znovuobjaviť princíp rozšírenia siete s párnym počtom vstupov o ďalšie dva vstupy prezentovaný v [Bid10], ktorý je ilustrovaný na obrázku 31. Pri rozširovaní 4-vstupovej siete o 4 vstupy sa podarilo znížiť hodnotu oneskorenia o takmer štvrtinu oproti spomínanému princípu získanému inštrukciami. Siete generované týmto vzorom zobrazuje obrázok 32.

	6	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**
konvenčne	15	28	45	66	91	120	153	190	231	276	325	378
inštrukcie	12	22	35	51	70	92	117	145	176	210	247	287
e2_n2_p1_s2_L	12	22	35	51	70	92	117	145	176	210	247	287
e4_n2_p3_s9_L	12	23	38	57	80	107	138	173	212	255	302	353
e4_n4_p9_s13_L		20		45		80		125		180		245
e4_n4_p10_s13_L		*19*		*42*		*74*		*115*		*165*		*224*
e4_n4_p11_s13_L		20		45		80		125		180		245

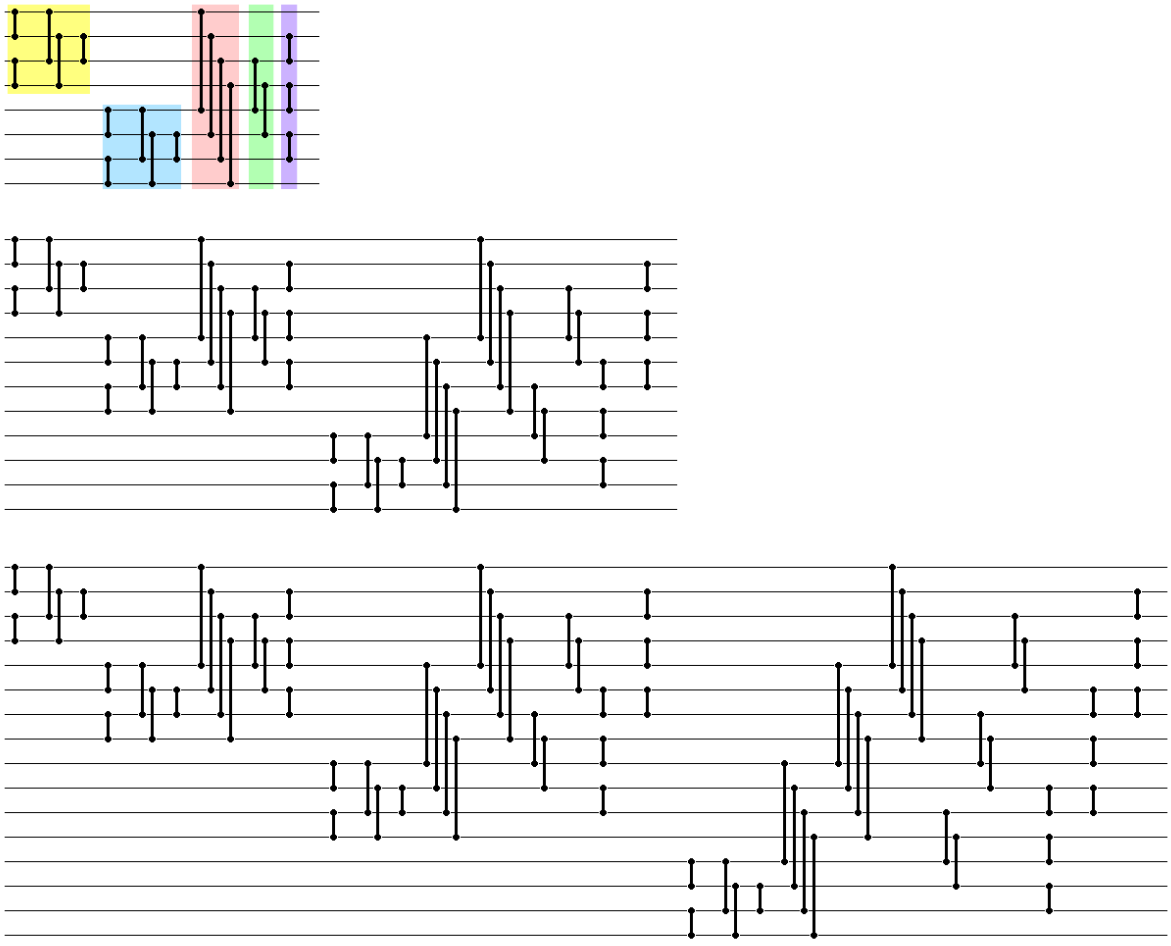
Tabuľka 8: *Počet komparátorov sietí s párnym počtom vstupov, ktoré vznikli opakovaním vrstiev získaného vzoru pri použití radiaceho bloku.*

	6	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**26**	**28**
konvenčne	9	13	17	21	25	29	33	37	41	45	49	53
inštrukcie	7	11	15	19	23	27	31	35	39	43	47	51
e2_n2_p1_s2_L	6	9	12	15	18	21	24	27	30	33	36	39
e4_n2_p3_s9_L	6	9	13	17	21	25	29	33	37	41	45	49
e4_n4_p9_s13_L		7		14		21		28		35		42
e4_n4_p10_s13_L		*6*		*11*		*16*		*21*		*26*		*31*
e4_n4_p11_s13_L		7		12		17		23		29		35

Tabuľka 9: *Oneskorenie sietí s párnym počtom vstupov, ktoré vznikli opakovaním vrstiev získaného vzoru pri použití radiaceho bloku.*



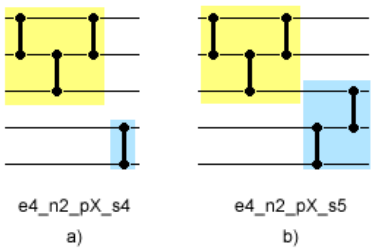
Obrázok 31: *Znovuobjavenie princípu rozširovania sietí o 2 vstupy prezentovaného v [Bid10] pomocou vzoru e2_n2_p1_s2_L.*



Obrázok 32: *Ukážka najlepšieho nájdeného vzoru e4_n4_p10_s13_L rozširujúceho 4-vstupové embryo o 4 vstupy.*

6.3 Návrh generických radiacich sietí s nepárnym počtom vstupov

Do tejto kategórie experimentov spadá rozširovanie 3-vstupového embrya o 2 vstupy Vhodné výsledky priniesla len metóda opakovania vzorov pri použití radiacich blokov. Pri ostaných metódach získané vzory buď generovali siete s nekvalitnými parametrami, alebo dokonca neplatné siete. Prehľad použitých radiacich blokov sa nachádza na obrázku 33.



Obrázok 33: *Použité radiace bloky pre 3-vstupové embryo a) e4_n2_pX_s4 b) e4_n2_pX_s5*

6.3.1 Experimenty s opakovaním vzorov pri použití radiaci blokov

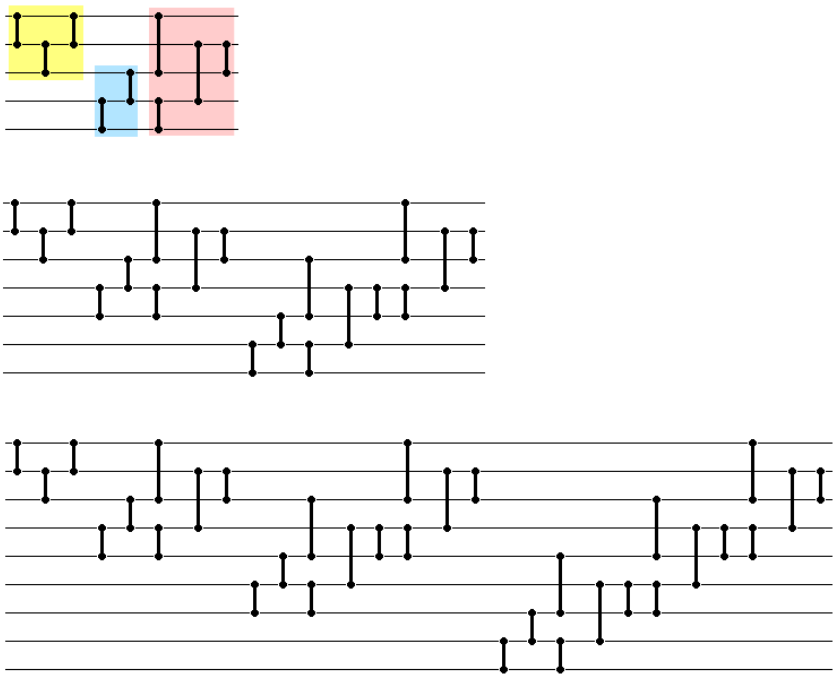
Pre rozšírenie 3-vstupového embrya o 2 vstupy bol nájdený vzor *e3_n2_p3_s5*, ktorý generuje siete s menším počtom komparátorov ako konvenčný prístup, ich oneskorenie je však väčšie. Radiace siete generované týmto vzorom zobrazuje obrázok 34. Najlepšie parametre, o sa týka počtu komparátorov, v tejto kategórii dosahujú siete získané pomocou inštrukcií.

	5	**7**	**9**	**11**	**13**	**15**	**17**	**19**	**21**	**23**	**25**	**27**
konvenčne	10	21	36	55	78	105	136	171	210	253	300	351
inštrukcie	12	22	35	51	70	92	117	145	176	210	247	287
e3_n2_p1_s4	9	20	36	57	83	114	150	191	237	288	344	405
e3_n2_p2_s5	10	22	39	61	88	120	157	199	246	298	355	417
e3_n2_p3_s5	*9*	*19*	*33*	*51*	*73*	*99*	*129*	*163*	*201*	*243*	*289*	*339*

Tabuľka 10: *Počet komparátorov sietí s nepárnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru pri použití radiaceho bloku.*

	5	7	9	11	13	15	17	19	21	23	25	27
konvenčne	7	11	15	19	23	27	31	35	39	43	47	51
inštrukcie	9	14	19	24	29	34	39	44	49	54	59	64
e3_n2_p1_s4	6	12	20	28	36	44	52	60	68	76	84	92
e3_n2_p2_s5	*6*	*11*	*17*	*23*	*29*	*35*	*41*	*47*	*53*	*59*	*65*	*71*
e3_n2_p3_s5	*6*	*11*	*17*	*23*	*29*	*35*	*41*	*47*	*53*	*59*	*65*	*71*

Tabuľka 11: *Oneskorenie sietí s nepárnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru pri použití radiaceho bloku.*

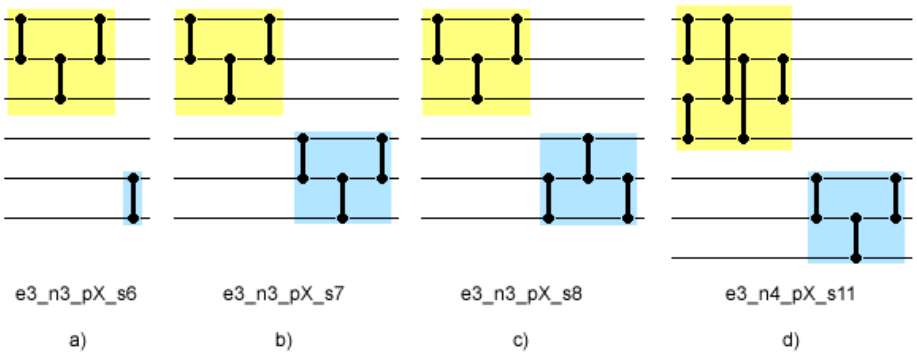


Obrázok 34: *Ukážka vzoru e3_n2_p3_s5 rozširujúceho 3-vstupové embryo o 2 vstupy.*

6.4 Návrh generických radiacich sietí striedavo s párnym a nepárnym počtom vstupov

Samostatnou oblasťou sú vzory, ktoré generujú striedavo párne a nepárne siete. Patria sem 3-vstupové a 4-vstupové embryo, ku ktorým pridávame 3 nové vstupy. Podobne ako v pri generovaní sietí s párnym počtom vstupov všetky metódy okrem opakovania vzorov bez použitia radiaceho bloku priniesli zaujímavé výsledky. Na obrázku 35 sa nachádza prehľad použitých radiacich blokov. Keďže sa výskum zameraný na návrh generických sietí nezaoberal sieťami so striedavo párnym a nepárnym počtom vstupov budú v tabuľkách tejto podkapitoly pre párne vstupy sietí uvedené najlepšie hodnoty

parametrov sietí získaných inštrukciami pre párny počet vstupov a pre nepárny počet vstupov hodnoty parametrov sietí získaných inštrukciami pre nepárny počet vstupov.



Obrázok 35: *Použité radiace bloky pre 3- a 4-vstupové embryo a) e3_n3_pX_s6 b) e3_n3_pX_s7 c) e3_n3_pX_s8 d) e3_n4_pX_s11*

6.4.1 Experimenty s opakovaním vzorov

Túto metódu sme aplikovali na získané vzory, pričom najlepšie parametre dosahovali 3-vstupové embryá rozšírené o 3 vstupy. Tabuľka 12 uvádza počty komparátorov pre siete vygenerované týmito vzormi, ktoré ale nedosiahli prevratné hodnoty. Menšie vylepšenie oproti inštrukciám je možné vidieť u nepárnych sietí v tabuľke 13, ktorá uvádza oneskorenia sietí, pri vzore *e3_n3_p3*, čo je však stále horšie ako konvenčný spôsob tvorenia sietí.

	6	**9**	**12**	**15**	**18**	**21**	**24**	**27**
konvenčne	15	36	66	105	153	210	276	351
inštrukcie	12	35	51	92	117	176	210	287
e3_n3_p1	13	33	63	103	153	213	283	363
e3_n3_p3	13	33	63	103	153	213	283	363

Tabuľka 12: *Počet komparátorov sietí so striedavo párnym a nepárnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru.*

	6	**9**	**12**	**15**	**18**	**21**	**24**	**27**
konvenčne	9	15	21	27	33	39	45	51
inštrukcie	7	19	19	34	31	49	43	64
e3_n3_p1	7	16	25	34	43	52	61	70
e3_n3_p3	7	15	23	31	39	47	55	63

Tabuľka 13: *Oneskorenie sietí so striedavo párnym a nepárnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru.*

6.4.2 Experimenty s opakovaním vzorov pri použití radiaci blokov

Zapojením radiacich blokov sme opäť dostali lepšie výsledky ako bez nich. Zmenšil sa počet komparátorov a zároveň sa znížilo oneskorenie generovaných sietí a to pre párny aj nepárny počet vstupov. Ako je vidno v tabuľke 14, pri vzore *e3_n3_p8_s8* stačilo 243 komparátorov pre 27-vstupovú radiacu sieť. Oneskorenie pri oboch prezentovaných vzoroch dosiahlo rovnakých hodnôt ako uvádza tabuľka 15.

	6	**9**	**12**	**15**	**18**	**21**	**24**	**27**
konvenčne	15	36	66	105	153	210	276	351
inštrukcie	12	35	51	92	117	176	210	287
e3_n3_p7_s7	13	30	54	85	123	168	220	279
e3_n3_p8_s8	*12*	*27*	*48*	*75*	*108*	*147*	*192*	*243*

Tabuľka 14: *Počet komparátorov sietí so striedavo párnym a nepárnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru pri použití radiaceho bloku.*

	6	**9**	**12**	**15**	**18**	**21**	**24**	**27**
konvenčne	9	15	21	27	33	39	45	51
inštrukcie	7	19	19	34	31	49	43	64
e3_n3_p7_s7	*6*	*12*	*18*	*24*	*30*	*36*	*42*	*48*
e3_n3_p8_s8	*6*	*12*	*18*	*24*	*30*	*36*	*42*	*48*

Tabuľka 15: *Oneskorenie sietí so striedavo párnym a nepárnym počtom vstupov, ktoré vznikli opakovaním získaného vzoru pri použití radiaceho bloku.*

6.4.3 Experimenty s opakovaním vrstiev vzorov pri použití radiacich blokov

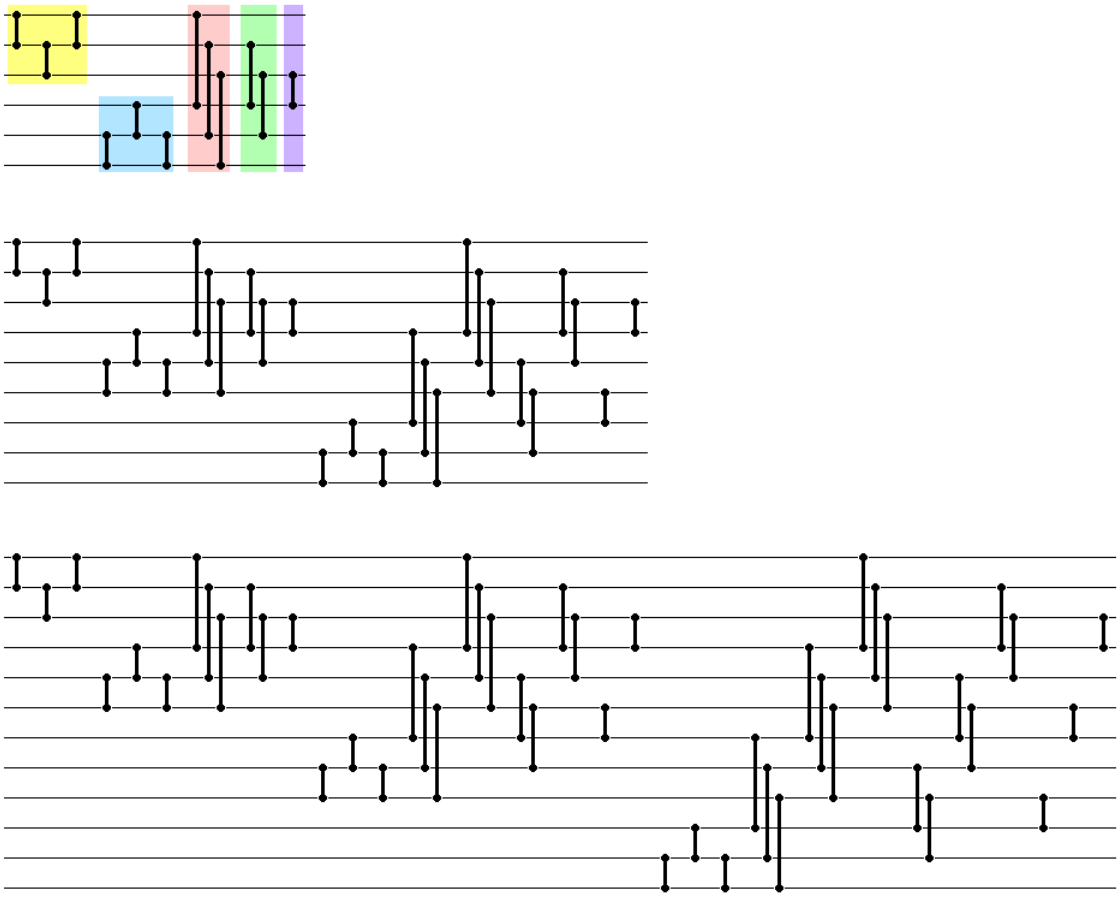
Opakovanie vrstiev získaných vzorov ešte o niečo zlepšilo parametre z kapitoly 6.4.2, čo dokazuje vzor *e3_n3_p8_s8_L*. Je lepší v oboch parametroch pre svoje párne aj nepárne siete ako najlepšie konvenčné riešenie a aj riešenie pomocou inštrukcií. Nie je síce lepší ako najlepší vzor *e4_n4_e10_s13_L*, ale pre siete, ktoré pomocou tohto vzoru nevygenerujeme je lepší ako vzor *e2_n2_p1_s2_L*. Počet komparátorov pre vzory z tejto metódy uvádza tabuľka 16, oneskorenie tabuľka 17. Obrázok 36 ilustruje najlepší získaný vzor v tejto oblasti ako aj siete ním generované.

	6	9	12	15	18	21	24	27
konvenčne	15	36	66	105	153	210	276	351
inštrukcie	12	35	51	92	117	176	210	287
e3_n3_p5_s6_L	13	32	60	97	143	198	262	335
e3_n3_p8_s8_L	*12*	*27*	*48*	*75*	*108*	*147*	*192*	*243*

Tabuľka 16: *Počet komparátorov sietí so striedavo párnym a nepárnym počtom vstupov, ktoré vznikli opakovaním vrstiev získaného vzoru pri použití radiaceho bloku.*

	6	9	12	15	18	21	24	27
konvenčne	9	15	21	27	33	39	45	51
inštrukcie	7	19	19	34	31	49	43	64
e3_n3_p5_s6_L	7	16	25	34	43	52	61	70
e3_n3_p8_s8_L	*6*	*10*	*14*	*18*	*22*	*26*	*30*	*34*

Tabuľka 17: *Oneskorenie sietí so striedavo párnym a nepárnym počtom vstupov, ktoré vznikli opakovaním vrstiev získaného vzoru pri použití radiaceho bloku.*



Obrázok 36: *Ukážka vzoru e3_n3_p8_s8_L rozširujúceho 3-vstupové embryo o 3 vstupy.*

Pri rozširovaní 4-vstupového embrya o 3 nové vstupu dostávame striedavo nepárne a párne radiace siete. Napriek väčšiemu počtu komparátorov generuje vzor *e4_n3_p5_s11L* siete s menším oneskorením ako konvenčné riešenie a riešenie získané pomocou inštrukcií, nedosahuje ale kvalít vzorov získaných pri návrhu generických sietí s párnym počtom vstupom. V tabuľkách 18 a 19 môžeme porovnať parametre sietí generovaných týmito vzormi.

	7	**10**	**13**	**16**	**19**	**22**	**25**	**28**
konvenčne	21	45	78	120	171	231	300	378
inštrukcie	22	35	70	92	145	176	247	287
e4_n3_p5_s11_L	*16*	*35*	*62*	*97*	*140*	*191*	*250*	*317*
e4_n3_p6_s11_L	17	38	68	107	155	212	278	353

Tabuľka 18: *Počet komparátorov sietí so striedavo nepárnym a párnym počtom vstupov, ktoré vznikli opakovaním vrstiev získaného vzoru pri použití radiaceho bloku.*

	7	**10**	**13**	**16**	**19**	**22**	**25**	**28**
konvenčne	11	17	23	29	35	41	47	53
inštrukcie	14	15	29	27	44	39	59	51
e4_n3_p5_s11_L	*6*	*12*	*18*	*24*	*30*	*36*	*42*	*48*
e4_n3_p6_s11_L	7	13	19	26	33	40	47	54

Tabuľka 19: *Oneskorenie sietí so striedavo nepárnym a párnym počtom vstupov, ktoré vznikli opakovaním vrstiev získaného vzoru pri použití radiaceho bloku.*

6.5 Sumár výsledkov

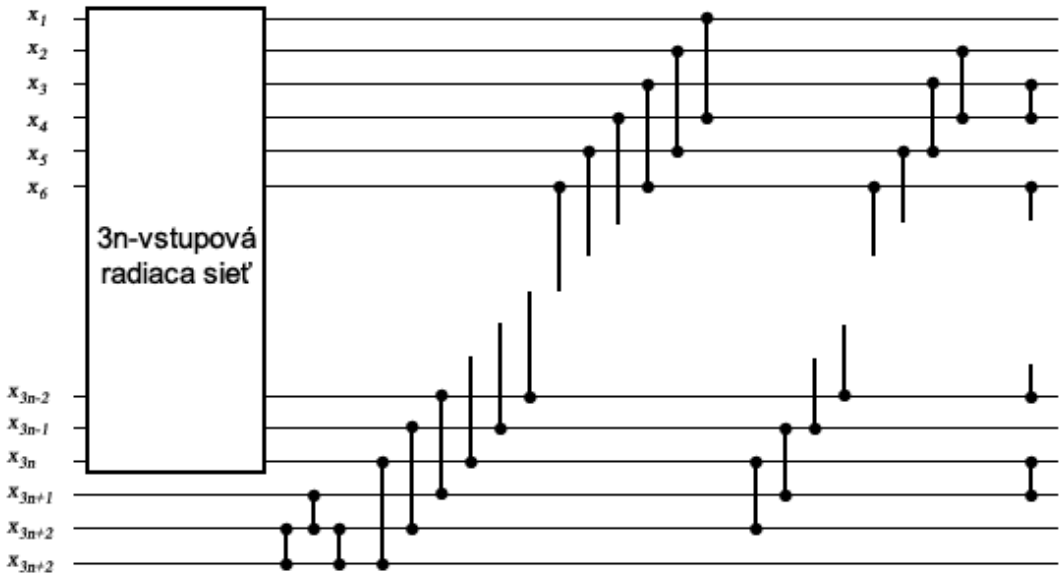
V rámci experimentov boli používané 4 metódy tvorenia generických radiacich sietí, z ktorých najlepšie výsledky dosahovala metóda opakovania vrstiev vzorov získaných z výpočtu celulárnych automatov. Vhodný vzor pre túto metódu sa nepodarilo nájsť pre generické siete s nepárnym počtom vstupov. Je pravdepodobné, že to môže byť dôvod prečo sme v tejto oblasti nenašli nové alebo nezopakovali už existujúce riešenia.

6.5.1 Princíp 3n+3

Jedno z inovatívnych riešení sa podarilo nájsť pre generické siete so striedavo párnym a nepárnym počtom vstupov, kedy počet vstupov rozširovanej radiacej siete je deliteľný číslom 3 (3*n*-vstupová radiaca sieť). Tento princíp sa nedá priamo porovnať s riešeniami získanými pomocou inštrukcií, pretože v tejto oblasti neboli vykonané žiadne známe experimenty. Oproti konvenčnému princípu potrebuje tento prístup pre 27 vstupov takmer o tretinu menej komparátorov a dosiahne presne

o tretinu menšie oneskorenie. Porovnanie závislostí počtu komparátorov od počtu vstupov princípu 3n+3 a konvenčného prístupu zobrazuje graf 1, porovnanie závislostí oneskorení od počtu ich vstupov graf 2. Náčrt tohto princípu ukazuje obrázok 37.

Výpočet počtu komparátorov popisuje rovnica 15, ktorá platí všeobecne. Oneskorenie novovzniknutej siete sa vypočíta rekurzívne podľa rovnice 16, ktorá platí pri generickom vytvorení z 3-vstupového embrya, pre ľubovoľnú 3*n*-vstupovú sieť sa najhoršie možné oneskorenie vypočíta podľa rovnice 17.

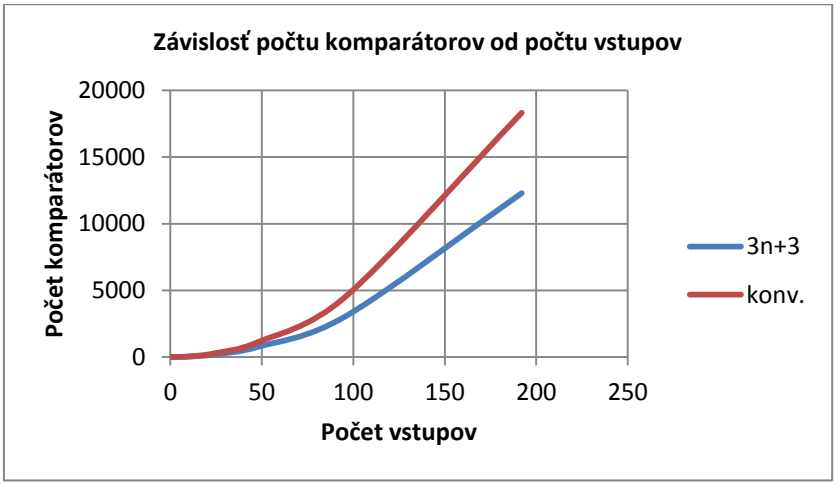


Obrázok 37: *Princíp tvorby 3n+3-vstupovej radiacej siete z existujúcej 3n-vstupovej radiacej siete.*

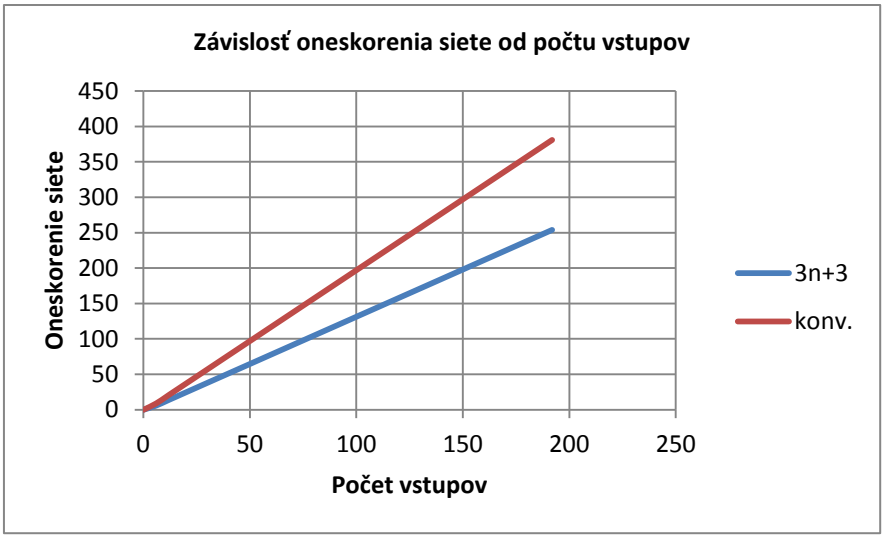
$$C(3n+3) = C(3n) + 6n + 3 \tag{15}$$

$$\begin{aligned} D(6) &= 6 \\ D(3n+3) &= D(3n) + 4 \end{aligned} \tag{16}$$

$$D(3n+3) = D(3n) + 2n + 1 \tag{17}$$

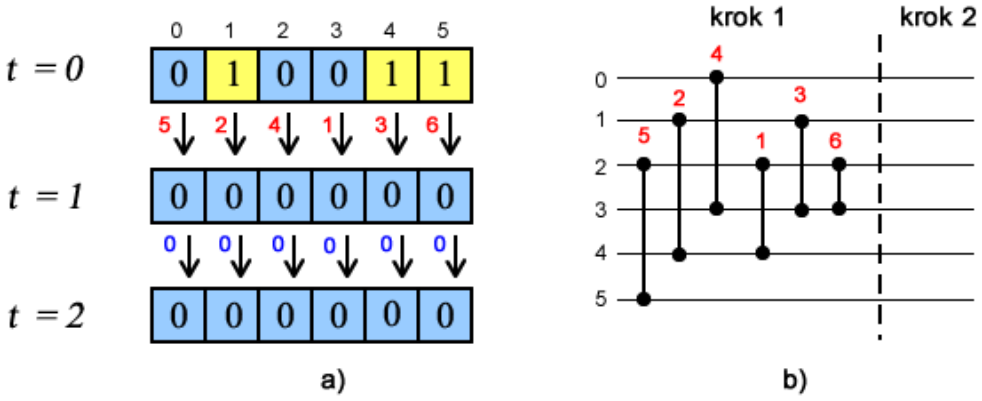


Graf 1: *Porovnanie závislostí počtu komparátorov od počtu vstupov konvenčného princípu a princípu 3n+3.*



Graf 2: *Porovnanie závislostí oneskorenia siete od počtu vstupov konvenčného princípu a princípu 3n+3.*

Celulárny automat, ktorý vygeneroval vzor *e3_n3_p8_s8_L*, pracoval s 2 stavmi a na získanie vzoru boli použité jeho 2 kroky, pričom vzor bol nájdený už po prvom kroku, po ktorom sa automat ustálil do homogénnej konfigurácie so samými nulami a negeneroval žiadne ďalšie platné komparátory. Automat začínal s konfiguráciou 010011 a používal pravidlá uvedené v tabuľke 20. Zmeny konfigurácií v jednotlivých krokoch spolu s vygenerovanými komparátormi uvádza obrázok 38, použité pravidlá pri zmene stavov buniek v týchto krokoch zase tabuľka 21.



Obrázok 38: *a) konfigurácie celulárneho automatu, ktorý vygeneroval b) vzor e3_n3_p8_s8_L. Čísla pravidiel ktoré generovali platné komparátory sú pri bunkách aj pri komparátoroch označené červenou farbou.*

Číslo pravidla	Pravidlo
0	000 → 0 : 2 2
1	001 → 0 : 2 4
2	010 → 0 : 1 4
3	011 → 0 : 1 3
4	100 → 0 : 0 3
5	101 → 0 : 2 5
6	110 → 0 : 2 3
7	111 → 1 : 0 5

Tabuľka 20: *Pravidlá celulárneho automatu, ktorý vygeneroval vzor e3_n3_p8_s8_L.*

Index CA	t = 0 → t = 1	t = 1 → t = 2
0	$101 \xrightarrow{5} 0 : 2\,5$	$000 \xrightarrow{0} 0 : 2\,2$
1	$010 \xrightarrow{2} 0 : 1\,4$	$000 \xrightarrow{0} 0 : 2\,2$
2	$100 \xrightarrow{4} 0 : 0\,3$	$000 \xrightarrow{0} 0 : 2\,2$
3	$001 \xrightarrow{1} 0 : 2\,4$	$000 \xrightarrow{0} 0 : 2\,2$
4	$011 \xrightarrow{3} 0 : 1\,3$	$000 \xrightarrow{0} 0 : 2\,2$
5	$110 \xrightarrow{6} 0 : 2\,3$	$000 \xrightarrow{0} 0 : 2\,2$

Tabuľka 21: *Pravidlá použité v jednotlivých krokoch výpočtu celulárneho automatu. Pri šípkach je označené číslo použitého pravidla.*

6.5.2 Princíp 4n+4

K ďalšiemu inovatívnemu riešeniu viedol vzor *e4_n4_p10_s13_L*, ktorý dosahoval najlepšie výsledky napriek tomu, že obsahoval nadbytočné komparátory. Po ich odstránení sa parametre sietí generovaných týmto vzorom ešte zlepšili a pre 28 vstupovú sieť dosiahli hodnoty 209 pre počet komparátorov a 28 pre oneskorenie, čo znamená zlepšenie takmer o polovicu pre oba parametre oproti konvenčnému princípu a takmer o tretinu pre oba parametre oproti princípu získanému pomocou inštrukcií. Porovnanie metód pre rôzne počty parametrov je vidieť v grafoch 3 a 4.

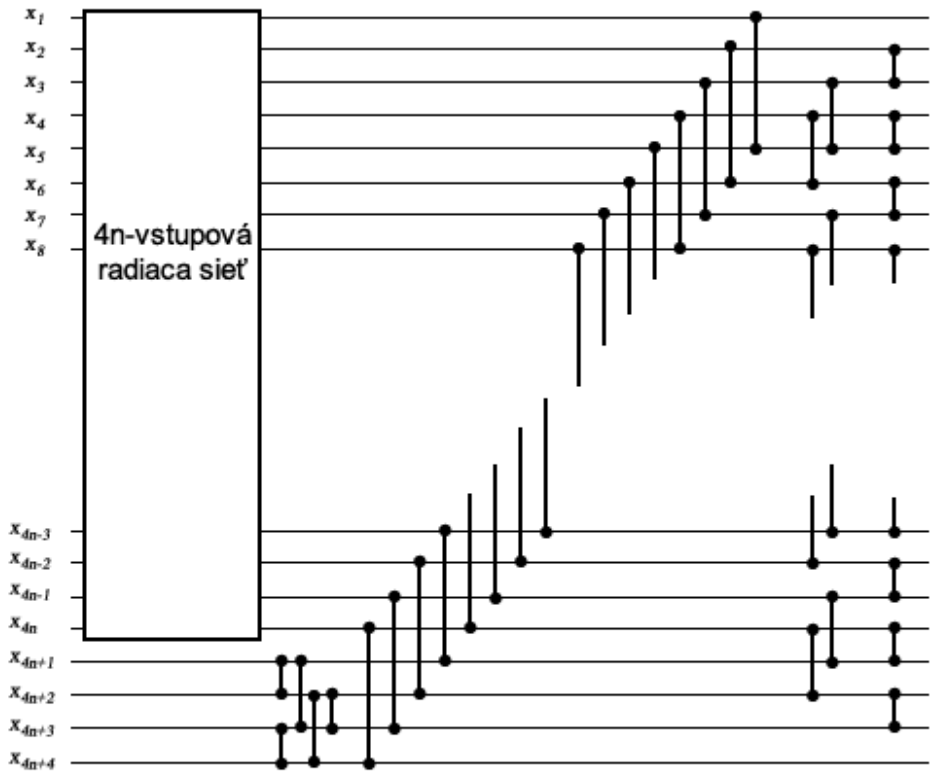
Počet komparátor ľubovoľne 4*n*-vstupovej siete rozšírenej týmto princípom získame podľa rovnice 17 a jej najhoršie možné oneskorenie dosiahne hodnotu z rovnice 19. Oneskorenie pri generickom vytvorení z 4-vstupového embrya získame zo vzorca 18. Princíp je ilustrovaný na obrázku 39.

$$C(4n+4) = C(4n) + 8n + 6 \tag{17}$$

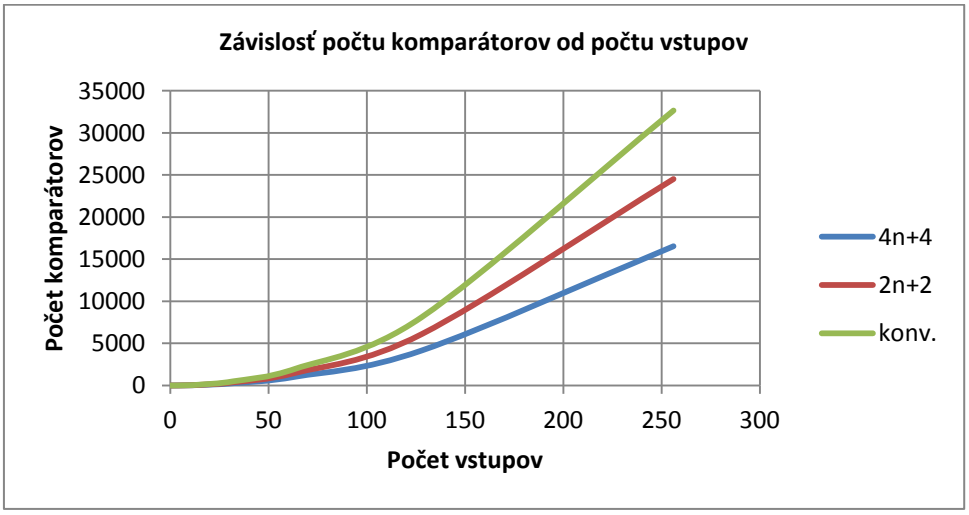
$$D(8) = 6$$

$$D(4n+4) = \begin{cases} D(4n) + 5, pre\ n = 3,5,7,9,\dots \\ D(4n) + 4, pre\ n = 2,4,6,8,\dots \end{cases} \tag{18}$$

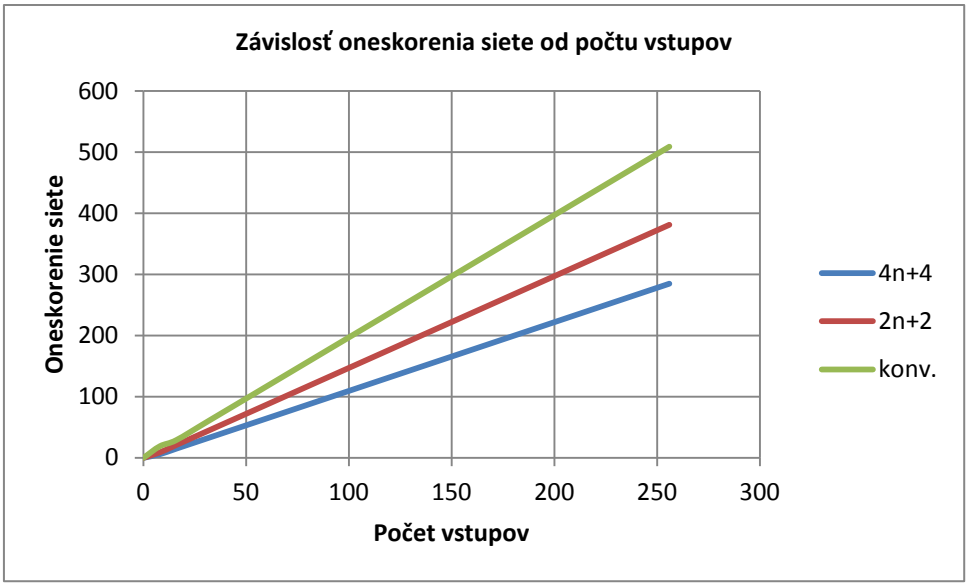
$$D(4n+4) = D(4n) + n + 2 \tag{19}$$



Obrázok 39: *Princíp tvorby 4n+4-vstupovej radiacej siete z existujúcej 4n-vstupovej siete.*

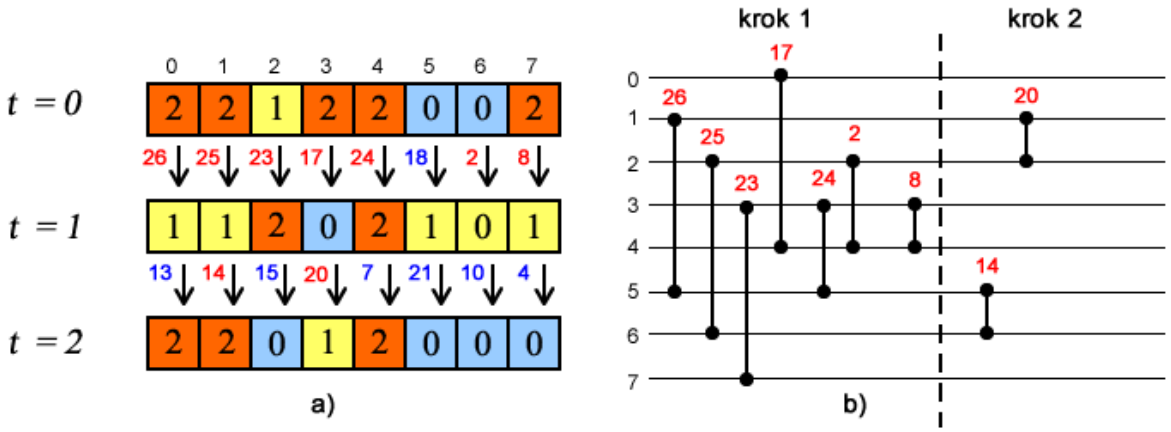


Graf 3: *Porovnanie závislostí počtu komparátorov od počtu vstupov konvenčného princípu a princípov 2n+2 a 4n+4.*



Graf 4: *Porovnanie závislostí oneskorenia siete od počtu vstupov konvenčného princípu a princípov 2n+2 a 4n+4.*

Celulárny automat, ktorý vygeneroval vzor *e4_n4_p10_s13_L*, pracoval s 3 stavmi a na získanie vzoru mu stačili 2 kroky z počiatočnej konfigurácie 22122002 a pravidlá uvedené v tabuľke 22. Zmeny konfigurácií v jednotlivých krokoch spolu s vygenerovanými komparátormi uvádza obrázok 40, použité pravidlá pri zmene stavov buniek v týchto krokoch zase tabuľka 23.



Obrázok 40: *a) konfigurácie celulárneho automatu, ktorý vygeneroval b) vzor e4_n4_p10_s13_L. Čísla pravidiel ktoré generovali platné komparátory sú pri bunkách aj pri komparátoroch označené červenou farbou.*

Číslo pravidla	Pravidlo	Číslo pravidla	Pravidlo	Číslo pravidla	Pravidlo
0	000 → 0 : 4 6	**9**	100 → 2 : 6 7	**18**	200 → 1 : 4 4
1	001 → 1 : 4 6	**10**	101 → 0 : 6 6	**19**	201 → 1 : 2 3
2	002 → 0 : 2 4	**11**	102 → 2 : 2 2	**20**	202 → 1 : 1 2
3	010 → 1 : 0 5	**12**	110 → 1 : 0 7	**21**	210 → 0 : 0 0
4	011 → 0 : 3 3	**13**	111 → 2 : 6 6	**22**	211 → 1 : 2 5
5	012 → 2 : 1 3	**14**	112 → 2 : 5 6	**23**	212 → 2 : 3 7
6	020 → 1 : 5 7	**15**	120 → 1 : 0 0	**24**	220 → 2 : 3 5
7	021 → 2 : 6 6	**16**	121 → 2 : 1 7	**25**	221 → 1 : 2 6
8	022 → 1 : 3 4	**17**	122 → 0 : 0 4	**26**	222 → 1 : 1 5

Tabuľka 22: *Pravidlá celulárneho automatu, ktorý vygeneroval vzor e4_n4_p10_s13_L.*

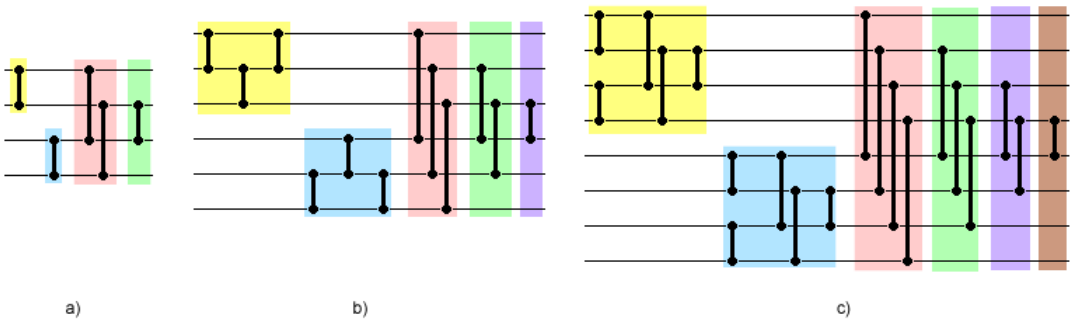
Index CA	t = 0 → t = 1	t = 1 → t = 2
0	$222 \xrightarrow{22} 1:1\,5$	$111 \xrightarrow{13} 2:6\,6$
1	$221 \xrightarrow{25} 1:2\,6$	$112 \xrightarrow{14} 2:5\,6$
2	$212 \xrightarrow{23} 2:3\,7$	$120 \xrightarrow{15} 1:0\,0$
3	$122 \xrightarrow{17} 0:0\,4$	$202 \xrightarrow{20} 1:1\,2$
4	$220 \xrightarrow{24} 2:3\,5$	$021 \xrightarrow{7} 2:6\,6$
5	$200 \xrightarrow{18} 1:4\,4$	$210 \xrightarrow{21} 0:0\,0$
6	$002 \xrightarrow{2} 0:2\,4$	$101 \xrightarrow{10} 0:6\,6$
7	$022 \xrightarrow{8} 1:3\,4$	$011 \xrightarrow{4} 0:3\,3$

Tabuľka 23: *Pravidlá použité v jednotlivých krokoch výpočtu celulárneho automatu. Pri šípkach je označené číslo použitého pravidla.*

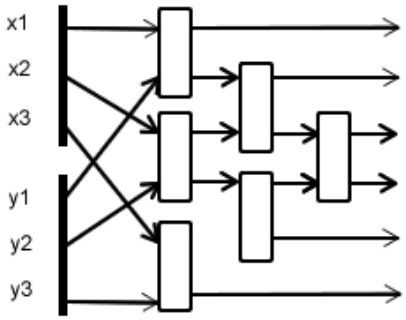
6.5.3 Princíp zotrieďovania dvoch n-prvkových neklesajúcich postupností

Pri skúmaní získaných vzorov sme objavili „vzor vo vzore“, konkrétne pri vzoroch *e2_n2_p1_s2_L*, *e3_n3_p8_s8_L* a *e4_n4_p10_s13_L*. Ak sa bližšie na tieto vzory pozrieme (obrázok 41), zistíme, že sa vždy jedná o rozšírenie *n*-vstupovej radiacej siete o *n* nových vstupov, ktoré sú následne radené podobnou sadou komparátorov. Samotný vzor slúži ako zotrieďovací blok (*merge*), ktorý zotrieďuje dve neklesajúce postupnosti do jednej neklesajúcej. Princíp tohto zotrieďovacieho bloku spočíva v tom, že porovnáva dvojice prvkov, ktoré majú vo svojich pôvodných postupnostiach rovnaký index.

Pre *n*-prvkové neklesajúce sekvencie *a* a *b* teda vytvorí dvojice $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$, ktoré porovná a z dvojice a_1, b_1 získa najmenší prvok a z dvojice a_n, b_n najväčší prvok z oboch postupností. Zo zostávajúcej sekvencii vytvorí nové dvojice na základe predchádzajúceho porovnania, získa druhý najmenší a druhý najväčší prvok výslednej postupnosti a rekurzívne pokračuje, až kým obe postupnosti nezotriedi do jednej neklesajúcej. Princíp zotrieďovacieho bloku pre dve 3-prvkové postupnosti ukazuje obrázok 42. Porovnanie parametrov so známymi zotrieďovacími algoritmami prinášajú tabuľky 24 a 25.



Obrázok 41: *"Vzor vo vzore" pre a) 2- b) 3-a c) 4-vstupové embryo.*



Obrázok 42: *Princíp zotrieďovacieho bloku.*

N	4	8	16	32	64	128	256
odd-even	5	19	63	191	543	1471	3839
bitonic	6	24	80	240	672	1792	4608
discovered	5	20	76	288	1140	4360	16912

Tabuľka 24: *Porovnanie počtu komparátorov zotrieďovacích princípov.*

N	4	8	16	32	64	128	256
odd-even	3	6	10	15	21	28	36
bitonic	3	6	10	15	21	28	36
discovered	3	7	15	31	63	127	255

Tabuľka 25: *Porovnanie oneskorení zotrieďovacích princípov.*

7 Záver

Genetické algoritmy sú jedným z príkladov, kedy sa človek snaží napodobniť procesy fungujúce v prírode a prispôsobiť ich svojim potrebám. Tieto algoritmy pomáhajú vedcom a iným výskumníkom nájsť riešenia tam, kde by to inak bolo náročné a zdĺhavé. Majú široké spektrum použitia, no napriek tomu, je potrebné venovať dostatočný čas ich aplikácii na konkrétny problém.

Návrh celulárnych automatov je jednou z oblastí, ktorá je z bežného hľadiska nie je jednoduchá a na jej automatizácii sa podieľajú práve evolučné techniky. V rámci tejto práce slúžili na návrh štruktúr, konkrétne boli ako štruktúry použité generické radiace siete s rôznym počtom vstupov. Na rozdiel od bežných, najmä grafických štruktúr reprezentovaných priamo v automate bol prevzatý model z [BVS10], ktorý upravuje pravidlá lokálnej prechodovej funkcie automatu tak, aby generovali radiacu sieť ako samostatnú štruktúru.

Celulárny automat v tomto prípade slúžil len ako generátor vzoru pre danú radiacu sieť (embryo) a samotná rozširujúca sa sieť vznikala aplikáciou metód opakovania vzorov alebo vrstiev vzorov prezentovaných v kapitole 5.

Veľké množstvo experimentov viedlo znovuobjaveniu princípu rozširovania 2*n*-vstupových radiacich sietí o 2 vstupy z [Bid10] a ďalej viedlo k objaveniu hneď niekoľkých inovatívnych riešení pre 3*n*-vstupové radiace siete rozširované o 3 nové vstupy, pre 4*n*-vstupové radiace siete rozširované o 4 vstupy a zotrieďovací blok na zotriedenie dvoch *n*-prvkových neklesajúcich postupností do jednej 2*n*-prvkovej neklesajúcej postupnosti. Podrobne sú tieto riešenia predstavené v podkapitole 6.5.

7.1 Zhodnotenie a nadväzujúca práca

Podobne ako v [Bid04] a [Bid10] sa nám podarilo pomocou genetického algoritmu získať riešenia, ktoré sú inovatívne a zároveň škálovateľné, čo pri evolučnom návrhu nie je bežným javom. Okrem nájdenia princípov na rozšírenie radiacich sietí o relatívne malý počet vstupov, bol objavený princíp, ktorý *n*-prvkovú sieť rozširuje o ďalších *n* vstupov, kde *n* je ľubovoľne veľké číslo (>1).

To môže byť inšpiráciou pri hľadaní vzorov pre zotrieďovanie ľubovoľne veľkých sekvencií, ktoré by dosahovali lepších parametrov ako algoritmy prezentované v podkapitole 4.2 Výskum sa samozrejme nemusí obmedzovať na doménu radiacich sietí, ale môže dosiahnuté výsledky aplikovať v rade ďalších oblastí zaoberajúcimi sa návrhom generických systémov

Literatúra

[Bat68] K. E. Batcher. Sorting networks and their applications. In Proc. *AFIPS Spring Joint Computer Conference*, zväzok 32, s. 307–314, April 1968.

[BCG82] E. R. Berlekamp, J. H. Conway, a R. K. Guy. *Winning Ways for your Mathematical Plays*, zväzok 2, kapitola 25, s 817–850. Academic Press, New York, 1982.

[Bid04] M. Bidlo. *Evoluční návrh řadicího algoritmu*, diplomová práce, Brno, FIT VUT v Brně, 2004

[Bid10] M. Bidlo. *Evolutionary Design of Generic Structures Using Instruction-Based Development*, Faculty of Information Technology BUT, Brno, 2010.

[BN62] R. C. Bose and R. J. Nelson. A sorting problem. *J. Assoc. Comput. Mach.*, s. 282–296, 1962.

[BSV10] M. Bidlo, K. Slaný, Z. Vašíček. Sorting Network Development Using Cellular Automata, In: *Evolvable Systems: From Biology to Hardware*, s. 85-96, London, GB, Springer, 2010.

[CVM07] J. Clegg , J. A. Walker , J. F. Miller, A new crossover technique for Cartesian genetic programming, *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007, London, England

[Dar06] C. Darwin. *On the Origin of Species By Means of Natural Selection*. Dover publications, 2006.

[FK73] R. W. F loyd and D. E. Knuth. The Bose-Nelson sorting problem. In J. N. Srivastava, editor, *A Surveyof Combinatorial Theory*. North-Holland, 1973.

[Gar70] M.Gardner, Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life", *Scientific American 223*, s. 120-123, 1970.

[GGK+03] A. Grama, A. Gupta, G. Karypis, V. Kumar: *Introduction to Parallel Computing*, Addison Wesley, 2003.

[HU79] J. E. Hopcroft a J. D. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley, Redwood City, CA, 1979.

[Hyn08] J. Hynek. *Genetické algoritmy a genetické programovanie*. Grada Publishing, Praha, 2008.

[Knu73] D. E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison Wesley, 1973.

[KPT00] V. Kvasnička, J. Pospíchal a P. Tiňo. *Evolučné algoritmy*. Vydavateľstvo STU, Bratislava, 2000.

[Lan84] C.G. Langton, Self-reproduction in cellular automata, *Physica. D* 10: 135–144, 1984.

[LB95] M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, kapitola 74, s. 5148–5150, June 1995.

[MCH94] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. *Evolving cellular automata to perform computations: Mechanisms and impediments*. Physica D, kapitola 75, s.361–391, 1994.

[Mun08] T. Munakata. *Fundamentals of the New Artificial Intelligence : Neural, Evolutionary, Fuzzy and More*. Springer-Verlag, London, 2008.

[Pac88] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, a M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, s. 293–301. World Scientific, Singapore, 1988.

[Par91a] I. Parberry. A computer-assisted optimal depth lower bound for nine-input sorting networks. *Mathematical Systems Theory*, kapitola 24, s. 101–116, 1991.

[Par91b] I. Parberry. On the computational complexity of optimal sorting network verification. In *Proceedings of The Conference on Parallel Architectures and Languages Europe*, in *Series Lecture Notes in Computer Science*, zväzok 506, s. 252–269. Springer-Verlag, 1991.

[Par92] I. Parberry, The Pairwise Sorting Network. Parallel *Processing Letters*, zväzok. 2, kapitoly 2,3, s. 205-211, 1992.

[SB05] L. Sekanina a M. Bidlo. *Evolutionary Design of Arbitrarily Large Sorting Networks Using Development*, Springer Science, 2005.

[SD08] S.N. Sivanandam a S.N. Deepa. *Introduction to Genetic Algorithms*. Spriger-Verlag, Berlin, 2008.

[Sip97a] M. Sipper. Evolving uniform and non-uniform cellular automata networks, in *Annual Reviews of Computational Physics*. D. Stauffer, Ed.,zväzok V, s. 243-285. World Scientific, Singapore, 1997.

[Sip97b] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, Heidelberg, 1997.

[Sip98] M. Sipper. Programming cellular machines by cellular programming, in *Bio-Inspired Computing Machines: Toward Novel Computational Architectures*. M. Tomassini and D. Mange, Eds., s. 99-119. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.

[ST98] M. Sipper and M. Tomassini. An introduction to cellular automata, in *Bio-Inspired Computing Machines: Toward Novel Computational Architectures*. M. Tomassini and D. Mange, Eds. , s. 49-58. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.

[vN66] J. von Neumann. *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[Wol94] S. Wolfram. *Cellular Automata and Complexity*. Addison-Wesley, Reading, MA, 1994.

Zoznam príloh

Príloha A. Obsah CD

Priložené CD má nasledovnú štruktúru:

`/program/sn/`	Program na evolúciu radiacich sietí.
`/program/snsupport/`	Podporné programy na optimalizáciu a zobrazenie radiacich sietí a program na tvorbu generických sietí zo získaných vzorov.
`/results/images/`	Obrázky získaných generických radiacich sietí.
`/results/parameters/`	Parametre získaných generických radiacich sietí.
`/text/dp_xbezak00.pdf`	Text diplomovej práce.

Príloha B. Inštalácia, konfigurácia a spúšťanie programov

V rámci tejto práce bolo implementovaných niekoľko programov v prostredí systému Linux. Program *sn* slúži na evolučný návrh radiacich sietí s pevným počtom vstupov ako aj na hľadanie vzorov pre už existujúce radiace siete (embryá) a na implementáciu bol použitý jazyk C/C++. Výstupom tohto programu je sekvencia komparátorov tvoriaca radiacu sieť s príslušnými parametrami ako aj počiatočná konfigurácia celulárneho automatu a sada získaných pravidiel. Nevýhodou programu je, že parametre treba nastaviť priamo v kóde, čo však neprekážalo pri našich experimentoch. Druhý program bol implementovaný v jazyku Python a využíva sa na tvorbu generických radiacich sietí z embrya a získaného vzoru. Oproti predchádzajúcemu je flexibilnejší a jeho parametre sa nastavujú cez príkazový riadok. Ďalšie pomocné programy implementované tiež v jazyku Python optimalizujú získané siete a dokážu ich previesť do grafickej podoby. Na tento prevod program používa knižnicu PIL, ktorá sa nenachádza na školskom serveri, preto na ňom nie je možné spustiť programy *snimage.py* ani *snpattern.py*.

B1. Program *sn*

Táto aplikácia generuje vzory pre generické radiace siete pomocou celulárneho automatu získaného evolúciou. V súbore *main.cpp* môžeme definovať niekoľko parametrov:

`int steps = 3`	Počet krokov automatu.
`int ca_size = 8;`	Veľkosť automatu.
`int states = 4;`	Počet stavov automatu.
`int comp_max_width = 1;`	Max. šírka komparátorov pri použití relatívneho kódovania.
`int encoding = ABSOLUTE;`	Kódovanie RELATIVE | ABSOLUTE
`float size_coef = 1000.0;`	Koeficient vplyvu počtu komparátorov na hodnotu fitness.
`float delay_coef = 1000.0;`	Koeficient vplyvu oneskorenia siete na hodnotu fitness

Embryo je možné definovať v súbore *ca.cpp* vo funkcii *prepareNetworkEmbryo()* pre každý komparátor nasledovným spôsobom (ukázaný je komparátor [0,1]):

```
tmp_comp.input_1 = 0; tmp_comp.input_2 = 1;
embryo->push_back(tmp_comp);
```

Program *sn* je potrebné po každom nastavení parametrov preložiť pomocou priloženého skriptu *make*.

Parametre genetického algoritmu sa dajú nastaviť v súbor *ga_settings* a po ich zmene nie je potrebný preklad programu.

Výstup programu je formátovaný ako jeden riadok *CSV* súboru, čo je vhodné pri automatickom spúšťaní a následnom spracovaní dát.

Každý komparátor výslednej siete je zapísaný nasledovne [x,y], kde *x* a *y* označujú indexy vodičov, na ktoré je komparátor napojený. Medzi indexmi, zátvorkami a čiarkou nie sú žiadne medzery a jednotlivé komparátory sú oddelené bielymi znakmi. Tento formát musí byť striktne dodržaný pre nasledujúce programy.

B2. Program *snpattern.py*

Program *snpattern.py* vytvára generické radiace siete podľa zvolenej metódy. Pracuje s niekoľkými vstupnými súbormi so sekvenciami komparátorov. Prvý súbor *sn_embryo* obsahuje embryo, druhý súbor *sn_pattern* vzor nájdený programom *sn* a tretí súbor *sn_sorter* môže obsahovať radiaci blok. Súbor s radiacim blokom je potrebné definovať pomocou príkazového riadku ako bude uvedené neskôr. Program po spustení bez parametrov a nájdení spomínaných súborov (okrem *sn_sorter*) používa metódu opakovania celých vzorov, vytvorí siete pomocou troch opakovaní, každú z nich otestuje a vypíše jej parametre. Na nastavenie metódy opakovania, formy výstupu prípadne zmeny vstupných súborov slúži príkazový riadok. Prehľad všetkých možných parametrov sa vypíše po spustení s prepínačom –h (--help):

```
Usage: ./snpattern.py [OPTIONS]

  -h, --help                  print this help
  -i, --images                output sorting networks as images
  -i, --text                  output sorting networks as text
  -l, --layers                use repeating layers to create sorting
                              network
  -x,                         turn of 1-0 principle evaluation
  -e file, --embryo=file      define an input file with embryo sorting
                              network
  -p file, --pattern=file     define an input file with pattern for
                              sorting network
  -s file, --sorter=file      define an input file with sorter for new
                              inputs
  -r number, --repeat=number  define number of repetitions
```

B3. Podporné programy

Prvý podporný program *snoptimal.py* vyhodnocuje pomocou *zero-one principle* postupnosť komparátorov zo štandardného vstupu vo formáte popísanom v časti B1. Každý komparátor, ktorý bol použitý je označený špeciálnou značkou (*flag*). Pri druhom prechode postupnosťou komparátorov sú na štandardný výstup v rovnakom formáte ako vstup odoslané len tie komparátory, ktoré majú túto značku nastavenú.

Druhý podporný program *snimage.py* slúži na prevod postupnosti komparátorov do grafickej podoby, konkrétne do formátu PNG. Vstupom programu je opäť popisovaná sekvencia komparátorov, ktorá je načítaná do vrstiev komparátorov. V rámci vrstvy sú komparátory dodatočne usporiadané vzostupne podľa indexu prvého komparátora. Ďalej sú vo vrstvách vytvorené skupiny komparátorov, ktoré budú vykreslené v jednej línii a následne dôjde k samotnému vykresleniu do súboru *sorting_network.png*.

Ukážková radiaca sieť je uvedená v súbore `/program/snsupport/sn_string` v priloženom CD. Možnosti použitia podporných programov s týmto súborom sú nasledovné:

```
./snoptimal.py < sn_string
./snimage.py < sn_string
./snoptimal.py < sn_string | ./snimage.py
```